
ATARI
ST
COMPUTER
Die Fachzeitschrift für den ATARI-ST Anwender.
Januar 90
DM 7,-
Ös. 56,-
Sfr. 7,-
1
Stacy
ATARIs
Laptop
TOS 1.4
Patches
1.44 MB
auf ST-Disketten



esprit erwartet von seinen Spielern ein gewisses Maß an Witz, Geist, Mut und Neugier. Eigenschaften, die auch im wirklichen Leben von einigem Vorteil sind. Man lasse sich also ein in ein Labyrinth aus ausgefeilten Hürden, falschen Hindernissen und (lösbaren) Rätseln. Schließlich ein Hinweis am Rande: **esprit** erklärt sich selbst, die Regeln werden sich finden. Fortgeschrittene Spieler erwartet mitten im Spiel noch ein Bonbon zum Knobeln. Ach: **esprit** gibt es jetzt beim Fachhändler.
esprit
APPLICATION SYSTEMS HEIDELBERG
Application Systems Heidelberg Software GmbH, Englerstraße 3, Postfach 10 26 46, D-6900 Heidelberg 1, Telefon (0 62 21) 30 00 02, Fax (0 62 21) 30 03 89 **in Österreich:** Reinhart Temmel Ges.m.b.H., Markt 109, A-5440 Golling, Telefon (0 62 44) 7081-0, Fax 7 18 83. **in der Schweiz:** DTZ DataTrade AG, Langstraße 94, CH-8021 Zürich, Telefon (01) 2 42 80 88, Fax 291 05 07.
MMAD Kommunikation

EDITORIAL

Liebe Leser und Leserinnen,

es ist nun schon vier Jahre her, daß die erste Ausgabe der ST Computer sich zum ersten Mal in den Regalen der Zeitungskioske und Händler präsentierte. Damals, der ATARI ST war gerade erst ein halbes Jahr auf dem Markt, gab es kaum Software (die ersten Programme waren größtenteils von anderen Rechnern portiert und kannten das Wort GEM nicht einmal vom Hörensagen.), kaum Hardware (512 kByte und fast nur einseitige Laufwerke waren beim alten 520ST Standard.) und schon gar keine Informationen. Es gab nur ein ATARI-Entwicklungspaket, das größtenteils aus der Original-Dokumentation von Digital Research für PCs bestand, also nur bedingt benutzbar war, da auf viele Interna des ST nicht eingegangen wurde.

Man war also ziemlich auf sich allein gestellt und ausprobieren und testen war der Alltag eines engagierten ST-Besitzers. Auch uns Redakteuren ging es nicht anders. An solche Meilensteine der ST-Programmierung wie Tempus, Signum!, Wordplus und STAD, um nur ein paar zu nennen, war gar nicht zu denken. So waren es neben den wichtigen Grundlagen-Artikeln, die in vielen Dingen die offiziellen Dokumentationen ergänzten, vor allem die Hardware-Basteleien in der ST Computer, die in der Anfangszeit immer wieder große Resonanz hervorriefen (Speicheraufrüstungen, Fremdlaufwerke etc.). Unter dem Stichwort "Back to the Roots" haben wir in dieser Jubiläumsausgabe auch wieder den Schwerpunkt auf Hardware-Basteleien gelegt, um zu zeigen, daß man auch nach vier Jahren so einiges mit seinem ST anstellen kann. Vielleicht bleiben Sie uns ja weitere vier Jahre treu.

1986 1987 1988 1989 1990

4 JAHRE ST COMPUTER

Ein frohes Fest und einen guten Rutsch in neue Jahr wünscht Ihnen Ihre ST Computer-Redaktion!

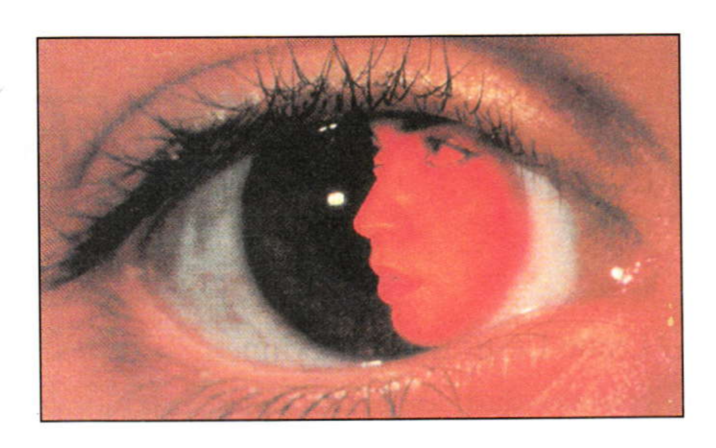
Harald Egel

I N H A L T

SOFTWARE

ANWENDUNGEN

GRUNDLAGEN



TmS Color Express Ein Bildverarbeitungssystem für ATARI ST

Der Markt für digitale Bildverarbeitung boomt. Professionelles DTP ohne hochwertiges Bildmaterial ist fast undenkbar. Mit Calamus können von Text- und Vektorgrafiken brillante Filme belichtet werden, aber Fotos wurden bisher nur mit einem sehr groben Raster, welches die Scanner selbst produziert haben, wiedergegeben. Für den Grafiker, der Bilder noch verändern, verfremden oder retuschieren will, gab es bisher kaum eine Möglichkeit, seinen Beruf auf dem ATARI ST zufriedenstellend auszuführen. Bisher arbeitete man auf dem Mac mit Graustufenbildbearbeitungen bzw. in 8-Bit-Farbgrafik. Erst seit kurzer Zeit arbeitet man auch dort mit 24 Bit, die ungefähr 16,8 Millionen Farben entsprechen. Das ist nun mit Color Express von TmSauch auf dem ST möglich.

Seite **16**

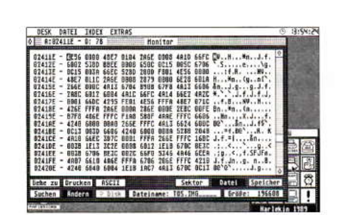


Ganz schön dicht - HD-Laufwerke am ST

Wieder fällt eine Barriere, und Sie erleben in dieser "ST-Computer" diese kleine Sensation mit: HD-Laufwerke können mit minimalem Aufwand am ST angeschlossen und betrieben werden. Auf die für ATs und PS/2-Rechner üblichen HD-Disketten passen 1.44 Megabytes (mit HYPERFORMAT sogar einiges mehr), der Datenaustausch mit PCs wird endgültig zum Kinderspiel.

Seite **28**

TOS





DATEN & PATCHES

Rund ums TOS

In dieser Ausgabe geht es gleich in zwei Beiträgen um das Intimste, was unser ST zu bieten hat, das Betriebsystem TOS. Im ersten Beitrag wird ein Programm vorgestellt, mit dem es möglich ist, u.a. die Versionsnummern der jeweiligen TOS-Teile (AES, GEMDOS usw.) zu bekommen. Nebenbei wird dabei gleich mal wieder gezeigt, wie man sich Daten durch "saubere" Programmierung beschafft.

Der zweite Artikel ist ganz dem Modifizieren von TOS-Version 1.4 gewidmet. Da man auch bei ATARI nicht fehlerfrei programmiert, stellen wir sechs nützliche Patches für die neue Betriebssystemversion vor, die dieser Tage jetzt endlich auf den Markt kommen soll. Für Mega ST-Besitzer wird noch der Umbau von Megabit- auf 256k-(EP)ROMs besprochen u.v.m.

Seite **122 & 131**



Man geht nicht ohne: STACY

Endlich ist er da, der ST-Laptop! Nachdem man ihn schon auf diversen Messen bewundern konnte, wird er jetzt in zwei Version ausgeliefert, weitere sollen folgen. Sicherlich ist er gegenüber den anderen ST-Modellen nicht ganz billig, er bietet aber im Vergleich zu andern Laptops dieser Preisklasse durchaus mehr. Lasen Sie sich von unseren ersten Erfahrungen mit STACY überraschen.

Seite **34**

HARDWARE

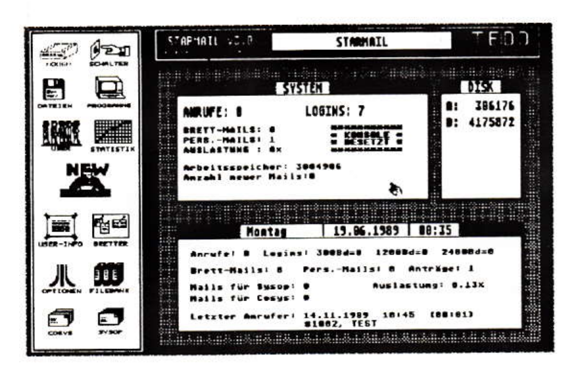
PROGRAMMIERPRAXIS

AKTUELLES

RUBRIKEN

NEWS

Starmail-Mailbox



Das Starmail-Mailbox-Programm wird jetzt in einer komplett überarbeiteten Version ausgeliefert. Die Sysop-Oberfläche ist nun von Menüleisten auf Icons umgestellt worden. Außerdem wurde eine ZModem-Routine eingebaut. Alle Programmversionen bis 2.5 werden nun unter dem Namen Starmail-Junior vertrieben. Die Versionen ab 3.0 laufen unter dem Namen Starmail-Master. Die Master-Version kostet DM 448,-; alle registrierten Benutzer können für DM 138,- ein Update anfordern.

TEDD Datentechnik
Gladiolenweg 19
4792 Bad-Lippspringe

Fujitsu-Drucker sind dokumentenecht

Die Fujitsu-Seitendrucker-Modelle RX7100, RX7100PS, RX7200m und RX7300E haben von der Bundesanstalt für Materialforschung und -prüfung das Zertifikat der Dokumentenechtheit erhalten. Die Drucker sind daher "...zur Herstellung von Schriftstücken in verschiedenen Bereichen des Urkundenwesens, z.B. für die Verwendung im Notariat zur Herstellung von Ausfertigungen und beglaubigten Abschriften im Sinne der Dienstordnung für Notare (Paragr. 27) geeignet...". Diese Bescheinigung ist damit zum ersten Mal einer gesamten Familie von Seitendruckern vom 5 Seiten/Minute bis zum Abteilungsdrucker mit 17 Seiten/Minute ausgestellt worden. Die Dokumentenechtheit wird nur in Verbindung von Toner, Drucker und Papiersorte bescheinigt. Die Drucker erhielten die Bescheinigung für Fujitsu-Originaltoner und Neusiedler-Kopierpapier HP'E weiß 80g/m2.

Fujitsu Deutschland GmbH
Rosenheimer Straße 145
8000 München 80
Tel. (089) 41301-0

Compu-Ware-News

Grafico, ein Accessory, ermöglicht es Ihnen, sämtliche Grafikformate, die es für den ATARI ST gibt, in die von Ihnen gewünschten und benötigten Dateiformate umzuwandeln. Eine Demoversion kostet DM 10,-, die Vollversion ist für DM 98,- erhältlich. Mit dem MIDI-MONITOR können Sie alle ankommenden Signale der MIDI-Schnittstelle anzeigen lassen. Dabei können Sie zwischen MIDI-Name, Hex- oder MIDI-Wert wählen. Das Programm läuft als Accessory und kostet DM 79,-. SCAN-PRINCE ist ein neuartiger Scanner. Mit welchem Programm Sie auch arbeiten, nutzen Sie Ihren Handyscanner. Die Software dafür läuft im Hintergrund mit. So ist es beispielsweise möglich, direkt aus 1st_Word, Signum!Zwei, Calamus etc. zu scannen. Scan-Prince läuft als Accessory und kostet DM 79,-

Source: DEGAS LoRes, DEGAS MidRes, DEGAS HiRes, COLORSTAR, MONOSTAR, NEOCHROME, IFF, ART DIRECTOR, STAD, DOODLE / 32K, IMG

Destination: DEGAS LoRes, DEGAS MidRes, DEGAS HiRes, COLORSTAR, MONOSTAR, NEOCHROME, IFF LoRes, IFF MidRes, IFF HiRes, ART DIRECTOR, STAD, DOODLE / 32K, IMG

TIME: 10:56:00
DATE: 05/11/89

GRAFIK-CONVERTER
VERSION 1.9 BETA
WRITTEN FOR
COMPU-WARE
by
PETER THIELE

START CONVERSION
SHOW PICTURE
PRINT PICTURE
SWAP PAGE
GO TO DESKTOP

COMPU-WARE
Dreufte 5
4250 Bottrop 2
Tel. (02045) 6302

Moleküldarstellung mit SIGNUM!Zwei

Bei mChem handelt es sich um ein Makropaket zu SIGNUM!Zwei. Damit ist es möglich, professionelle Formelbilder auf dem ATARI ST zu erstellen. Da die Makros nach dem Baukastensystem arbeiten, lassen sich die meisten Moleküldarstellungen mühelos erzeugen. Der Lieferumfang besteht aus 3 Fonts für 24-Nadler und Laser, 4 Makrosätzen und Handbuch. mChem kostet DM 99,- und ist zu beziehen über

Martin Frank
Bachstraße 18
6900 Heidelberg
Tel. (06221) 411541

4000 Grafiken

Eine Grafiksammlung mit 4000 Grafiken wird jetzt von der Firma Reinhard Rückemann Software aus Wuppertal angeboten. Die Grafiksammlung wird auf insgesamt 8 Disketten ausgeliefert, die je 500 Grafiken enthalten. Auf jeder Diskette befindet sich ein Convert-Programm, das die Anzeige der Bilder und die Übertragung in alle gängigen Grafikformate erlaubt. Mit den Disketten wird auch ein rund 200seitiger Katalog ausgeliefert, in dem alle Grafiken noch einmal ausgedruckt sind. Der Preis für die komplette Sammlung beträgt DM 139,- plus DM 5,- Versandkosten. Interessenten können auch eine Probediskette mit 500 Bildern und den vollständigen Katalog für DM 30,- plus DM 5,- Versandkosten anfordern. Dieser Preis wird bei einer Bestellung der kompletten Sammlung voll angerechnet.

RRSoft
Grundstraße 63
5600 Wuppertal 22
Tel. (0202) 640389

GADGET-Soundsampler

Der GADGET-Soundsampler hat ein Zusatzmodul bekommen. Der "LIVE-PLAYER" ermöglicht nun auch die Steuerung der Sampling-Ausgabe über ein MIDI-Keyboard. Im "LIVE-PLAYER" werden mit GADGET erstellte Teilstücke einfach den Tasten des MIDI-Keyboards zugewiesen. Weiterhin hinzugekommen ist das "BIGDISPLAY". Mit ihm wird das Zusammenschneiden der gesampleten Töne erheblich erleichtert. GADGET wird nunmehr mit den kommentierten Quelltexten der Assembler-Ausgaberoutinen geliefert. Das GADGET-Gesamtkonzept mit Soft- und Hardware kostet DM 298,-, registrierte Kunden zahlen für das Software-Update DM 20,-.

Sophisticated Applications
Computertechnik GbR
Friedrich-Ebert-Allee 2
2870 Delmenhorst
Tel. (04221) 14608

Print Q

Einen Drucker-Spooler bietet die Firma Bojan Smojver aus Jugoslawien an. Das Programm "Print Q" kann bis zu 15 Programme in eine Druckerwarteschlange stellen. Dabei kann für jede Datei angegeben werden, mit welchen Attributen sie gedruckt werden soll (NLQ, Fettdruck etc.). Außerdem kann der Spooler den Text links-, rechtsbündig oder im Blocksatz formatieren. Das Programm arbeitet mit verschiedenen Programmen zusammen und wurde beispielsweise unter Turbo C getestet.

Bojan Smojver
F. Lakusa 46
41260 Sesvete
Jugoslawien
Tel. (041) 264-324

fibuMAN EuroVersion 4.0

Es ist soweit. Die fibuMAN EuroVersion 4.0 ist zur Auslieferung bereit. Die Bezeichnung EuroVersion wurde gewählt, weil es mit diesem fibuMAN deutschsprachigen Anwendern möglich sein wird, Konten nach einem Kontenrahmen ihrer Wahl einzugeben. Jeder landesinterne Kontenrahmen, der mit bis zu 5stelligen Kontennummern arbeitet, wird dadurch unterstützt. Doch nicht nur Konten, auch Einnahme-Überschuß-Rechnung, Bilanz und Gewinn- und Verlustrechnung sind frei zu definieren, so daß Auswertungstexte entweder ganz oder teilweise aus den Vorgaben übernommen oder auch teilweise verändert werden können.

novoPLAN Software GmbH
Hardtstraße 21
4784 Rüthen 3
Tel. (02952) 8080
Fax: (02952) 3236

CADja-Version 1.2

Das CAD-Programm "CADja" liegt nun in der neuen Version 1.2 vor, die für DM 998,- bzw. DM 300,- (Update-Kosten) bezogen werden kann. So können der Maßstab jetzt von 1/500 bis 100/1 angegeben und Linien mit Kreisbögen verbunden werden, die Schraffur wird schneller ausgeführt, Bemaßungen können kopiert, Texte nachträglich geändert werden, die Linienstärke wird am Bildschirm dargestellt und vieles mehr. Die neue Version ist erhältlich bei

Computer Technik Kieckbusch
Baumstammhaus
5419 Vielbach
Tel. (02626) 78336
Fax: (02626) 78337

Creator-Update

Die Creator-Version 1.0 geriet sehr schnell an die Speichergrenzen eines 1MB-Systems. Deshalb wurde das Programm intensiv überarbeitet. Die neue Version ist mehr als 100 kB kürzer als ihr Vorgänger. Dadurch wird auch die Betriebssicherheit um einiges erhöht, denn viele Funktionen von Creator arbeiten sehr speicherintensiv. Gleichzeitig wurde der Druckertreiber so modifiziert, daß er mit dem geringstmöglichen Speicher arbeitet. Durch den optimierten Code ist ein deutlicher Geschwindigkeitszuwachs - im Bereich der Animation bis zu 100% - realisiert worden.

Application Systems /// Heidelberg
Englerstraße 3
6900 Heidelberg
Tel. (06221) 300002
Fax: (06221) 300389

DTP- und MIDI-Center

Die Frankfurter Firma Eickmann Computer erweitert Verkaufsräume und Angebot. Nach der Eröffnung von neuen Verkaufs- und Beratungsräumen in direkter Nachbarschaft des bisherigen Ladengeschäfts wird dort ab Anfang November weit mehr als die normale Beratung angeboten. So finden für interessierte Kunden wöchentliche DTP- (Desktop Publishing) und MIDI-Vorführungen statt. Hier können sich Interessierte über die Möglichkeiten des ATARI-Einsatzes im professionellen Grafik-Design- und Verlagsbereich sowie auf dem Musiksektor (MIDI) informieren und beraten lassen. Über diesen, unseres Wissens im Frankfurter Raum einmaligen, Service hinaus werden Schulungen für Anwender in den genannten Bereichen angeboten.

Eickmann Computer
In der Römerstadt 249 und 253
6000 Frankfurt/Main 90 - Praunheim
Tel. (069) 763409

Postzugelassenes Laptop-Modem

Nord Computer & Software präsentierte in München zwei neue Modems. Das "Bitwalker 9600 T V.32" ist für 9600 Baud ausgelegt. Die Postzulassung wird für die erste Novemberwoche erwartet. Das Modem unterstützt synchron und asynchron V.21, V.21bis, V.22 und V.32 und bietet neben dem CCITT-Standard auch den Hayes-AT-Befehlssatz. Das zweite Modem ist die postzugelassene Laptop-Modemkarte "Bitwalker 2400" mit maximal 2400 Baud nach V.22bis. Das Bitwalker 9600 T V.32 kostet DM 4550,-, für das Bitwalker 2400 müssen 1349,- DM bezahlt werden.

Nord Computer & Software
Emil-Kurz-Straße 1
8045 Ismaning
Tel. (089) 967527

Adimens-Prog für Turbo C

Der Markt&Technik Verlag AG stellt jetzt eine individuelle Anwendungsprogrammierung auf dem Datenbank-Kern von Adimens ST für den Turbo C-Compiler vor. PROG, das Zusatzmodul zum Datenbanksystem, verwaltet Ihre Daten nach dem relationalen Modell, während Sie sich auf Ihre Anwendung konzentrieren. Das Anlegen von Datendstrukturen geschieht mit dem Modul INIT. Ein grafischer Top-down-Entwurf übernimmt daberi die Anlage von record-Strukturen, Deklarationen und erforderlichen Header-Files. Adimens-Prog kostet 199,- DM (unverbindliche Preisempfehlung) und ist zu beziehen über

Markt&Technik Verlag AG
Buch- und Software-Verlag
Hans-Pinsel-Straße 2
8013 Haar bei München
Tel. (089) 4613-621

BASiC-HART

Neben der Tabellenkalkulation BASiCALC vertreibt die Münchner Firma POINT Computer mit BASiCHART nun eine erweiterte Version des Spreadsheets. Das Programm wurde um einen Grafikteil ergänzt und ermöglicht dem Benutzer nun neben dem Erstellen von Rechenblättern auch die sofortige Umsetzung seiner Zahlenwerte. Präsentationsgrafiken mit Linien, Symbolen, Treppen, Flächen, Balken oder 3D-Grafiken lassen sich ebenso realisieren wie Kuchen- oder Kreisdiagramme. Der ATARI-Laserdrucker wird ebenso unterstützt wie Großbildschirm oder Farbmonitor. BASiCHART kostet DM 198,-, registrierte Benutzer können bei POINT ein Upgrade für DM 100,- erwerben.

POINT Computer GmbH
Gollierstraße 70
8000 München 2
Tel. (089) 505657

Videoverwaltungsprogramm

Aus Österreich kommt ein Programm, mit dem sich Video-Filme verwalten lassen. Sortierte Listen, Suchfunktionen, Statistiken und viele Funktionen mehr sind mit diesem Programm kein Problem mehr. "MGL-Video" kostet 44,- DM und ist erhältlich bei

Mag. G. Lugmair
Lustkandlg. 48/13
A-1090 Wien

Zeitmaschine



Eine Einschaltverzögerung für den Computer, die der Festplatte Zeit läßt, sich einzulaufen, ist jetzt für 59,- DM erhältlich. Dabei ist kein Eingriff in den Rechner notwendig, sondern das Gerät wird einfach auf die Steckdose aufgesteckt. Die Zeit, um die das Einschalten des Hauptrechners verzögert werden soll, läßt sich variabel zwischen 10 und 60 Sekunden einstellen.

Lions Hard Systems GbR
Lutherstraße 23
5810 Witten
Tel. (02302) 84521

Neuer SUPRA-Distributor

Mit Wirkung vom 1. Oktober hat die Firma MAKRO C.D.E mit Sitz in Großwallstadt den alleinigen Vertrieb der SUPRA-Produktpalette in Europa übernommen. Die Produktpalette umfaßt z.Z. Festplattensysteme (intern, extern), Modems (intern, extern), Wechselfloppy, Wechselplatten und Videodigitzer.

MAKRO C.D.E
Schillerring 19
8751 Großwallstadt
Tel. (06022) 24405

ATARI-News

Rechtzeitig zur Systems bot auch ATARI einige Neuigkeiten an. WordPerfect ist eine der leistungsfähigsten Textverarbeitungen auf PC/AT, Macintosh, VAX, ATARI ST, AMIGA, UNIX etc. Die vorliegende Version auf dem ATARI ST entspricht in den Textfeatures vollständig den aktuellen Versionen 4.2 auf den anderen Computersystemen. Das Programm kostet DM 799,-. Neu im Vertriebsprogramm ist auch "Lavadraw Plus", ein pixelorientiertes Zeichen- und Bildverarbeitungsprogramm. Mit dem Programm können unter anderem alle gängigen Matrix- und Laserdrucker angesteuert werden. Für die neueste Version, Lavadraw Plus, empfiehlt ATARI seinen Fachhändlern einen Preis von 149,- DM. Bereits in der letzten Ausgabe berichteten wir über die neueste Version von 1st_Wordplus. Das Programm mit der Versionsnummer 3.15 ist für DM 249,- erhältlich. Die Update-Version ist für DM 129,- zu bekommen.

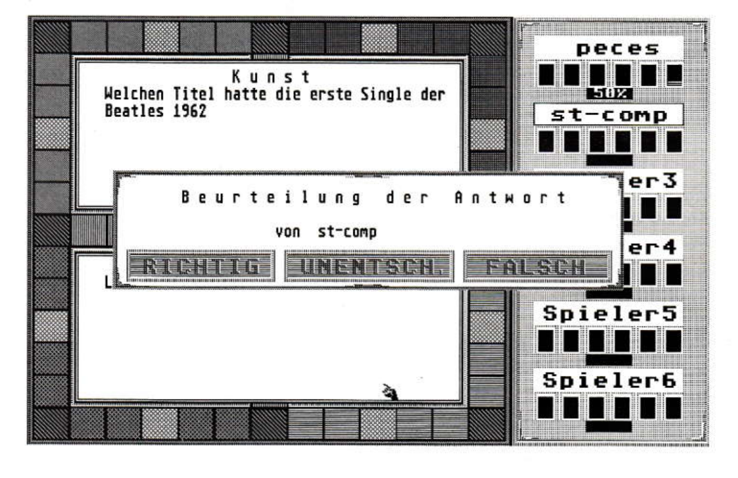
Bei allen Fachhändlern

ATARI
Frankfurter Straße 89-91
6096 Raunheim
Tel. (06142) 209-0

Modem-Test in Ausgabe 11

Leider ist und in der letzten Ausgabe ein kleiner Fehler unterlaufen. Auf Seite 20 hieß es, nur 3 Modems würden im Autoanswer-Modus 300, 1200 und 2400 Baud beherrschen. Wie Sie der Tabelle auf Seite 22 entnehmen können, beherrscht diesen Modus natürlich auch das postzugelassene Elsa MikroLink 2400 T2-Modem, so daß auch dieses zum Betrieb an einer Mailbox überaus geeignet ist. Wir bitten, diesen Fehler zu entschuldigen.

Megaquiz



Das Programm "Megaquiz" ist eine Mischung aus "Trivial Pursuit" und "Mensch ärgere dich nicht", welches auf allen ATARI ST mit TOS im ROM und 720 kB-Laufwerk läuft. Das Spiel funktioniert sowohl in Farbe, als auch in Schwarzweiß. Da die Diskette nicht kopiergeschützt ist, kann der Inhalt auch auf die Festplatte kopiert werden. Megaquiz kostet DM 29,90 auf doppelseitig formatierter Diskette und kann bezogen werden über

Roland Schnellberger
Edhof 9
8333 Hebertsfelden
Tel. (08721) 8519

CW-Chart 7.0

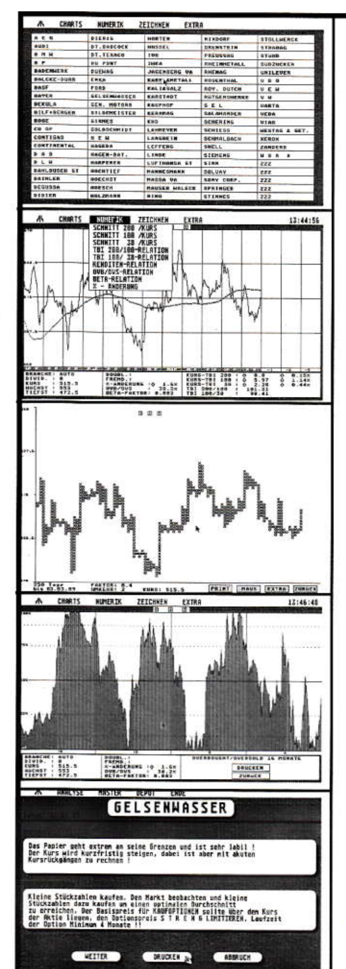
In erweitertem Umfang präsentiert sich in diesen Tagen die neue Version 7.0 der Börsensoftware CW-CHART/CW-DEPOT für den ST. Neue Speichertechniken führen zu Geschwindigkeitssteigerungen, ein neuer Compiler beschleunigt alle Berechnungen um 25%. Das Angebot an Charts und Auswertungsmöglichkeiten wurde unter anderem um die Funktionen Parabolic-Chart und -Tabelle, On-Balance-Volume, Spreads, RSI nach Wilder, Analyse des Umsatzverhaltens in der Umsatztabelle, Verknüpfung aller Tageskurse, Umsätze, Open Interest, Rel. Stärke im Line&Bar-Chart etc. erwwitert. CWCHART/DEPOT 7.0 kostet DM 997,-. Die Anwender der Vollversion 6.0 können ein Upgrade für DM 149,- erhalten. Auch eine Demo-Version kann für DM 20,- bezogen werden über

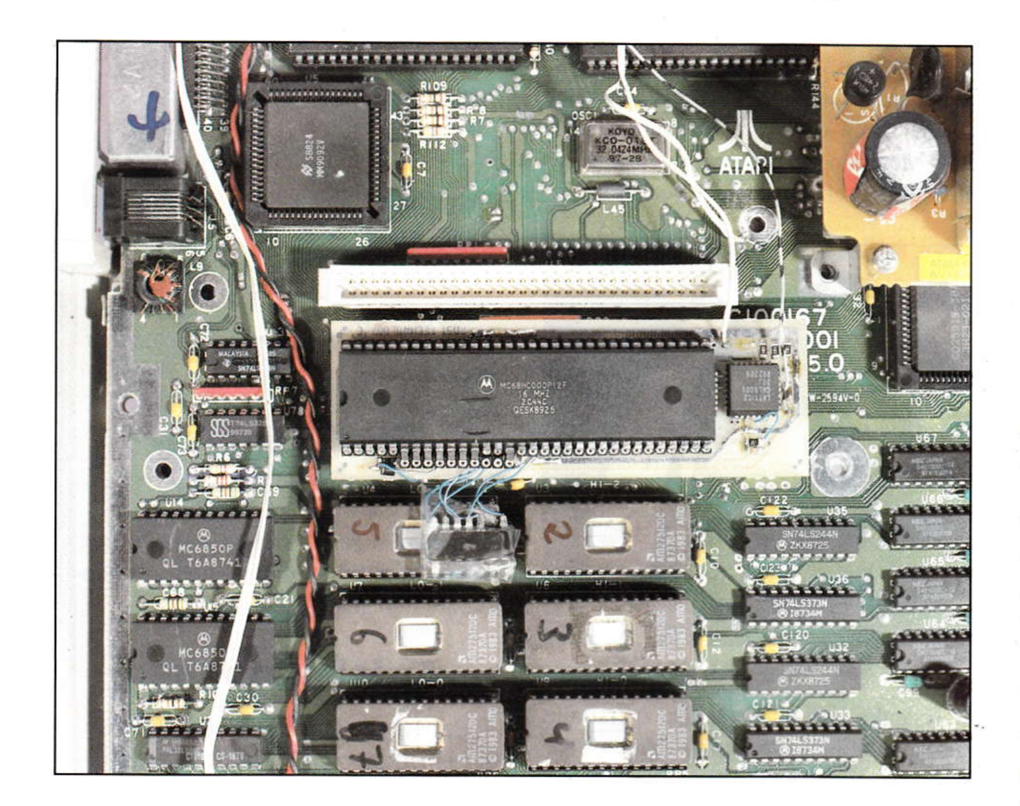
FOXWARE
Buchsteinweg 1
8172 Lenggries
Tel. (08042) 2175

Book One

Animierte Grafiken und Schautafeln zur Präsentation können nun auch mit einem "Original"-ATARI-Programm hergestellt werden. Mit "Book One" entwirft man direkt am Bildschirm seine Schautafeln, Grafiken und Texte. All diese Elemente können dann zu Filmsequenzen animiert werden. Digitalisierte Bilder können ebenso einbezogen werden wie freie Textformulierungen, die beispielsweise als Ergebnis einer Diskussion direkt abgespeichert werden. dadurch wird nur noch an einem Medium gearbeitet, egal ob Film, Standbild, Grafik, Textchart und Schreibtafel: alles ist im Computer. Für Book One empfiehlt ATARI seinen Händlern einen Verkaufspreis von DM 599,-.

Erhältlich bei allen
ATARI-Fachhändlern





Formel 1-ST

16 MHz-Takt im ATARI ST

Als im Juli Hypercache mit einem 16 MHz-Prozessor auf den Markt kam, waren viele ST-Anwender ganz aus dem Häuschen ob der Geschwindigkeitsvorteile, die diese Erweiterung bot. Ein schnellerer Prozessor ist schließlich eine sinnvolle Erweiterung. Nun geht das zweite Produkt ins Rennen: Turbo 16 von der Frankfurter Firma Eickmann Computer.

Der ST läßt sich gut erweitern - mit mehr Speicher, höheren Diskettenformaten, größeren Monitoren, Grafikkarten und vielem mehr. Warum also nicht den Prozessortakt verdoppeln? Ein "normaler" ST tuckert mit 8 MHz vor sich hin. Durch Turbo 16 wird der Takt verdoppelt, denn auf einer kleinen Zusatzplatine ist ein 68000-Prozessor enthalten, der mit 16 MHz arbeitet. So wird aus dem Trabbi-ST ein Porsche-ST. Neben dem Prozessor wird der ST aber auch noch durch einen 16 kB-Cache-Speicher erweitert, der viele Arbeitsvorgänge im Speicher um ein Vielfaches erhöht.

Auf- und Einbau

Leider, und das ist bei fast allen Erweiterungen so, geht es auch bei Turbo 16 nicht ohne Löten. Der alte Prozessor hat ausgedient und muß seinem großen Bruder weichen. Dazu muß er entweder ausgelötet oder mit einer Zange entfernt werden, denn er befindet sich leider direkt auf der Hauptplatine. Auf den ursprünglichen Prozessorplatz wird ein Sockel gelötet, auf den dann die Zusatzplatine aufgesetzt wird. Schnell sind noch einige Drähte angelötet, um den Cache-Speicher an- und ausschalten zu können und den 16 MHz-Takt zu gewinnen. Der Prozessor ist übrigens ein CMOS-Baustein, wird also nicht so warm wie ein "normaler" Prozessor. Dadurch wird er beispielsweise auch für Grafikkarten interessant, die ja eine recht beträchtliche Eigenwärme entwickeln. Nachdem der Einbau geschafft ist, kann man sich zurücklehnen und den ST anschalten. Nun sollte nur noch das mitgelieferte Accessory installiert werden, mit dem sich der ST wieder jungfräulich auf 8 MHz zurücksetzen läßt. Auch der Cache-Speicher läßt sich hier softwaremäßig abschalten.

Wie schnell?

Im Gegensatz zu einem 68020-Prozessor, auf dem nicht alle Programme laufen würden, verrichtet hier der gleiche Prozessor wie immer seinen Dienst. Dadurch können kaum Kompatibilitätsprobleme auftreten. Ansonsten ist, bis auf den Cache-Speicher, alles beim alten geblieben, also sollten auch kaum Programme existieren, die nicht funktionieren, sofern sie "sauber" programmiert sind. Ich habe verschiedene Programme ausprobiert, alle sind anstandslos gelaufen. Auch kritische haben den Test überstanden, sowohl ST als auch Programme erfreuen sich bester Gesundheit.

Bei der Frage nach der Geschwindigkeitssteigerung wirft sich meine Stirn in Falten, denn diese Frage ist nicht ohne weiteres zu beantworten. Die Erhöhung der Prozessortaktrate implementiert nicht zwingend eine Geschwindigkeitssteigerung bei allen Programmen. Der Prozessor selbst hat sich nicht geändert und benötigt für jeden Befehl die gleiche Anzahl von Taktzyklen wie sein kleiner Bruder. Diese werden lediglich doppelt so schnell abgearbeitet.

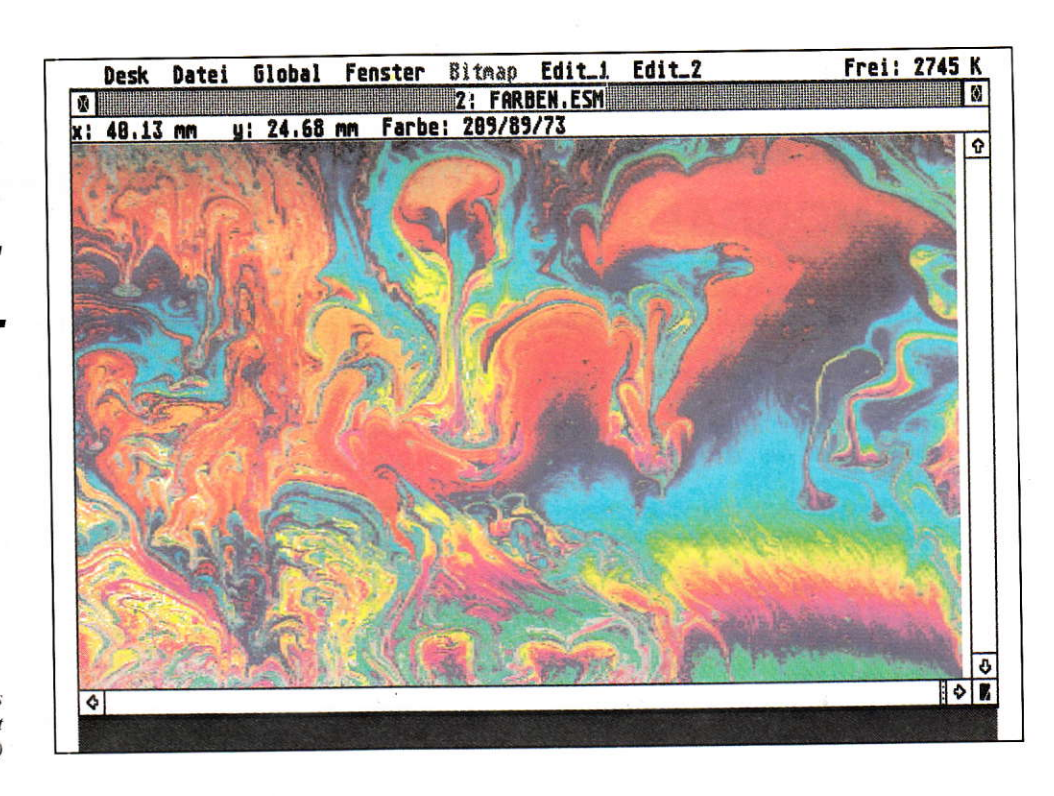
Cache

Normale, sogenannte CISK-Prozessoren haben den Nachteil, daß alle Befehle und Daten auf dem gleichen Platz liegen und über einen gemeinsamen Bus beim Prozessor ankommen. Der Prozessor ist ja nun auch nicht der schnellste und arbeitet alles der Reihe nach ab. Wenn der Bus belegt ist, muß halt jemand warten, bis er wieder frei ist. Dieses Problem hat man mit einem kleinen, aber feinen und sehr schnellen Cache-Speicher gelöst, der beispielsweise im 68020 und 68030 bereits serienmäßig eingebaut ist. Bei Turbo 16 ist der Cache-Speicher 16 k groß. Das ist eigentlich schon fast mehr als genug und beschleunigt den ST ungemein.

Turbo 16 stellt dem Prozessor ebenfalls einen Cache-Speicher zur Verfügung, der Einfachkeit halber allerdings extern. Dadurch ist gewährleistet, daß der Ablauf (fast) aller Programme beschleunigt wird, wenn auch nicht gerade um 100 Prozent. Nun habe ich aber immer noch nicht die Frage nach der Geschwindigkeitssteigerung beantwortet. Hier kann ein ganz leichter Grundsatz festgestellt werden.

TmS Color Express

Ein Bildverarbeitungssystem für ATARI ST



Eine Farbkopie des Bildschirms (MGE mit Color Express)

Der Markt für digitale Bildverarbeitung boomt. Professionelles DTP ohne hochwertiges Bildmaterial ist fast undenkbar. Mit Calamus können von Text- und Vektorgrafiken brillante Filme belichtet werden, aber Fotos wurden bisher nur mit einem sehr groben Raster, welches die Scanner selbst produziert haben, wiedergegeben.

Für den Grafiker, der Bilder noch verändern, verfremden oder retuschieren will, gab es bisher keine Möglichkeit, seinen Beruf auf dem ATARI ST zufriedenstellend auszuführen. Bisher arbeitete man auf dem Mac mit Graustufenbildbearbeitungen bzw. in 8-Bit-Farbgrafik. Erst seit kurzer Zeit arbeitet man auch dort mit 24 Bit, die ungefähr 16,8 Millionen Farben entsprechen. Das ist nun auch auf dem ST möglich, denn mit Color Express von TmS und (optional) einer MGE-Grafikkarte werden neue Maßstäbe gesetzt.

Für wen?

Die Programmkonzeption wurde sowohl für Laien als auch für professionelle Grafiker ausgelegt. Alle Arbeiten, die bisher noch mit einer "echten Dunkelkammer" erledigt wurden, kann Color Express auf dem ST machen. Aber das ist noch nicht alles. Es können Bildmanipulationen vorgenommen werden, die ansonsten nur mit größtem fototechnischem Aufwand zu realisieren waren. Da Color Express sowohl Bitmap-, Grauton- als auch Farbbilder bearbeiten kann, verfügt es über eine große Anzahl von Formaten, die es laden und speichern kann (Tabelle 1). Intern arbeitet Color Express nur mit 3 Bildarten. Zum ersten gibt es die gewohnte Bitmap, als monochrome Bilder mit einem Bit pro Bildpunkt, wie sie von Raster-Scannern erzeugt werden. Zum zweiten arbeitet Color Express mit Grautonbildern mit bis zu 256 Graustufen. Wurden diese Bilder mit einem Farb-Scanner oder einer Farbvideokamera eingescannt, erfolgt deren Bearbeitung intern sogar mit 768 Graustufen. Graustufenbilder besitzen pro Bildpunkt nicht nur die Information Weiß oder Schwarz, sondern je nach Format bis zu 256 verschiedene Helligkeitswerte. Die dritte Bildart haben wir die 24-Bit-True-Color-Bilder, mit denen bis zu 16,8 Millionen Farben dargestellt werden können. Das menschliche Auge kann nur ca. 6 Bit pro Farbe, insgesamt also 18 Bit unterscheiden, wodurch die 24 Bit genügend Farben für höchste Ansprüche zur Verfügung stellen.

Ladbare Formate:

TIFF
GEM-Image
Degas
Neochrome
IFF
Simplex
Extended Simplex
GFA Block
Screen-Formate wie Doodle etc.

Speicherbare Formate:

TIFF
GEM-Image
Degas
Neochrome
IFF
Simplex
Extended Simplex

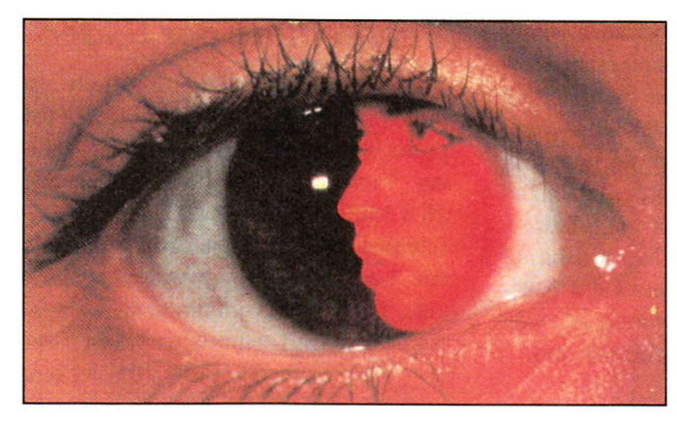
Tabelle 1: Lade- und Speicherformat von Color Express

Monitore

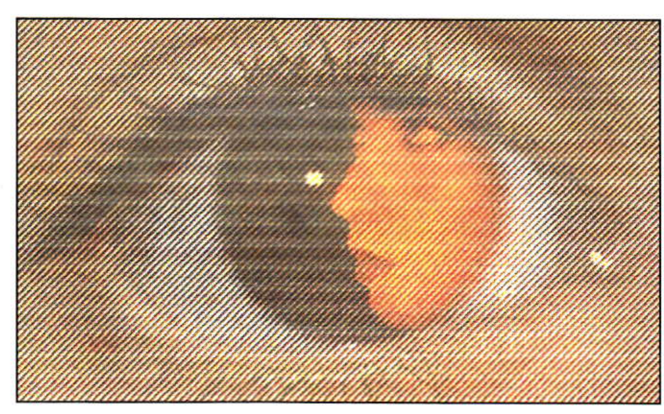
Besitzer eines monochromen Monitors sehen die Grauton- und Farbbilder in geditherter Form auf dem Bildschirm.

Dithering ist ein Verfahren, bei dem Grauton- bzw. Farbinformationen ausschließlich durch weiße und schwarze Bildpunkte wiedergegeben werden. Das Ergebnis ist ein fein gepunktetes Raster, das verschiedene Grauwerte durch mehr oder weniger gesetze Bildpunkte wiedergibt. So kann die Bildbearbeitung auch an monochromen Monitoren erfolgen. Grafikkarten wie die MGE von MAXON erleichtern die Arbeit ungemein, da schon am Bildschirm die Grauwerte bzw. Farben beurteilt werden können.

Color Express verfügt über ein modulares Konzept. Sämtliche Scanner- und Druckertreiber liegen als Accessories vor, die je nach Bedarf in den Arbeitsspeicher geholt werden können. Das trifft auch für spezielle Bildbearbeitungs-Tools zu. Eigene Druckertreiber für seltene Geräte der Anwendungen in der Industrie bzw. in der Lehre an Universitäten können dadurch leicht in das Programm integriert werden.



Ein Ausdruck auf einem 24-Nadeldrucker (NEC P6 Color)



Hier der gleiche Ausdruck mit einem Tintenstrahldrucker.



Ein Auge wurde eingescannt und in der rechten Hälfte mit einem farbverfremdeten Gesicht überlagert. Das TmS-Logo wurde mit dem Vektorzeichenprogramm TmS Graphics erzeugt und in Color Express eingeladen.

Welche Scanner?

Color Express arbeitet mit vielen Scannern zusammen, beispielsweise auch mit Geräten wie HAWK CP14, SPAT, Panasonic FX RS505, Microtek etc. Mit diesen Scannern kann man Color Express als Bitmap-Scanner benutzen. Dadurch ist nicht jeder Besitzer eines Raster-Scanners, der digitale Bildverarbeitung durchführen möchte, dazu verdammt, seinen teuren Scanner in den Ruhestand zu verbannen, um sich einen Grauton-Scanner zu kaufen.

Grauton-Scanner besitzen neben dem Raster- noch den Grauton- oder Multivalue-Modus und scannen pro Bildpunkt bis zu 256 Graustufen ein. Dabei sind Graustufen-Scanner nicht unerschwinglich. Geräte mit 64 echten Graustufen und 450 dpi, wie der Datacopy 730GS, sind für ungefähr DM 3000,- zu haben. Absolute Spitzen-Scanner wie der HP Scanjet Plus mit bis zu 600 dpi sind ein wenig teurer (DM 5300,-).

Wurde für den Scan-Vorgang ein Farb-Scanner verwendet, können für die Grautonberechnung 768 Graustufen genutzt werden, so daß auch schwierigste Bilder (z.B. schwarze Schrift auf rotem Grund) optimale Ergebnisse liefern. Die anschließende Ausgabe erfolgt dann mit den optimierten 256 Graustufen.

Farbbilder

Je nach verwendetem Scanner bzw. Videokamera werden bis zu 16,8 Millionen Farben eingelesen. Besitzt das Eingabegerät weniger Farben, wird intern auf 24 Bit umgerechnet. Da nicht alle Bildformate diese Farbenfülle speichern können, werden je nach Bedarf die Farben stufenlos auf bis zu 2 Farben oder Graustufen reduziert.

Color-Look-Up-Tabelle (CLUT): Color Express arbeitet als einziges ST-Programm permanent mit einer CLUT. Die Gamma-Korrektur wird erst auf ausdrücklichen Wunsch des Anwenders ins Bild übernommen. Diese Übernahme ins Bild bedeutet, daß unter Umständen der Kontrast, die Helligkeit oder die Farbzuordnung unwiderruflich geändert werden. Das bedeutet, daß auch nach längeren Arbeiten die ursprünglichen Farbwerte wiedererhalten werden können. Das Menü kann aber noch mehr: Der Kontrast kann verstärkt oder abgeschwächt werden, ebenso die Helligkeit. Mit dem sogenannten Histogramm läßt sich die Farbverteilung feststellen, so daß die Manipulationen der CLUT gezielt vorgenommen werden können. Natürlich sind noch weitaus mehr Manipulationen ausführbar, die jedoch unmöglich alle beschrieben werden können.

Masken...

...und Bildintervalle können unabhängig voneinander gewählt werden. Maske ist eine Funktion, mit der bestimmte Bildbereiche vor Veränderungen geschützt werden können. Die Maske kann direkt in Color Express erzeugt und/oder verändert

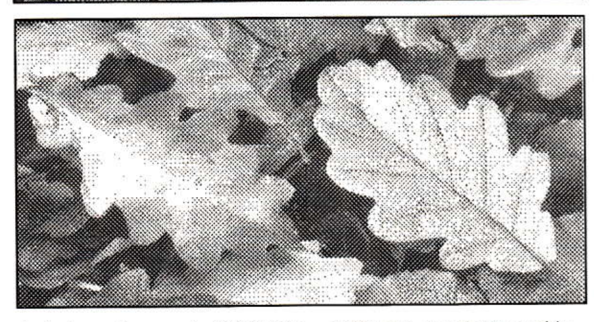
werden. Es lassen sich aber auch Bilder als Masken laden, was mitunter sehr interessante Effekte ergibt. Das Farbintervall beschreibt einen Farb- oder Grautonbereich, der nicht verändert werden darf. Damit lassen sich Bildpartien einer bestimmten Farbe wirkungsvoll schützen. Diese Option kann beispielweise dann angewendet werden, wenn etwa das Anlegen einer Maske zu schwierig ist, weil sehr kleine Bereiche zu schützen sind.



Auf einem HAWK CP14 wurde die Vorlage mit 200 dpi und 16 Graustufen eingelesen und anschließend mit Color Express in ein Graustufenbild umgerechnet.







Auf einem Panasonic FX-RS505 mit 200 dpi eingelesen und in ein Graustufenbild gewandelt.
Im oberen Bild eine Bitmap, in der Mitte Bitmap/Grautonbild, unten ein Grautonbild.

Bearbeitung der Bilder

Verschiedene Werkzeuge, die teilweise ihre Analogie in der herkömmlichen Reprotechnik haben, stehen dem Anwender zur Verfügung. Die bereits erwähnten 16,8 Millionen Farben können mit verschiedenen Werkzeugen bearbeitet werden. Lackfarbe beispielsweise überdeckt eine Bildpartie vollkommen, während Kohle erst nach mehrmaligem Überstreichen der Bildpartie deckend wird. Mit der Funktion "Aquarell" lassen sich die gleichen Effekte erzielen wie mit der "normalen" Aquarellmalerei.

Stift und Pinsel können mit allen drei Farbarten malen. Der Stift behält seine Form unbeschränkt bei, während der Pinsel je nach den eingestellten Werten seine Farbe verliert. Für beide Werkzeuge können die Farben aus einer Farbwahlbox, oder auch direkt aus dem Bild aufgenommen werden.

Bei Aufhellern und Abdunklern werden, je nach eingestelltem Wirkungsgrad, die überstrichenen Bildbereiche aufgehellt bzw. abgedunkelt. Dadurch lassen sich falsch belichtete Bereiche nachträglich in ihrer Helligkeit verändern.

Vorteilhaft ist, daß ein Undo-Puffer existiert. Wenn ein Bild in diesem Puffer vorliegt, können die alten Werte in das aktuelle Bild zurückgeholt werden, ohne daß alle Veränderungen verloren sind. Durch geschickten Einsatz dieses Tools lassen sich interessante Effekte erzielen.

Mit einem Werkzeug namens "Finger" lassen sich die Farben in den Bildbereichen, die er überfährt, verwischen, ohne sie zu vermischen. Hat man also harte Konturen in einem Bild, lassen sich diese problemlos mit dem Finger verwischen.

Wie es auch in der normalen Aquarellmalerei möglich ist, kann man bei Color Express je nach Intensität die Farben eines Bilds vermischen. Auch mit einem Stempel läßt sich retuschieren. Mit ihm werden Bildausschnitte aufgenommen und an anderer Stelle wieder eingefügt. Besonders bei der Retusche schwieriger Bildpartien, wie es z.B. bei der Retusche eines Gesichts vorkommt, lassen sich mit dem Pinsel sehr gute Ergebnisse erzielen.

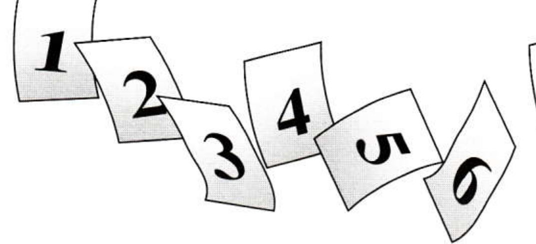
Collage-Möglichkeiten

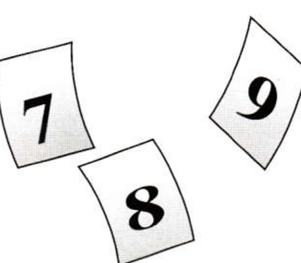
Was im Desktop des ATARI vergessen wurde, ist bei Color Express selbstverständlich eingebaut: Bildbereiche lassen sich auch mit unregelmäßigen Umrissen selektieren. Wird die Fenstergrenze erreicht, scrollt das Bild weiter. Dadurch gibt es bei der Selektion eines Bildausschnitts keine Größenbeschränkung. Die ausgewählten Bereiche lassen sich in das gleiche oder ein beliebiges anderes Bild integrieren. Dabei können verschiedene Operationen durchgeführt werden, deren Ergebnis direkt am Bildschirm sichtbar wird und die sofort nach Bedarf korrigiert werden können. Der zu kopierende Bereich kann stufenlos vergrößert und verkleinert werden. Durch nichtproportionales Ändern der x- und y-Koordinaten läßt sich das Bild linear verzerren. Neu für den ST sind auch nicht-lineare Verzerrungen. Dazu müssen nur eine oder alle Ecken des Bilds angewählt werden, worauf man das Bild nach Herzenslust zerren und biegen kann. Natürlich können die Bildausschnitte auch stufenlos gedreht werden. Sollten dann die Farben des kopierten Bereichs nicht mit denen des Zielbilds harmonieren, kann man durch "automatischen Farbangleich" die Farben automatisch anpassen. Schließlich kann noch ausgewählt werden, ob der zu kopierende Bereich den Zielbereich, oder beide Bilder sich prozentual überlagern sollen. Alle hier erwähnten Operationen arbeiten wahlweise unter Beachtung der Maske und des Farbintervalls.

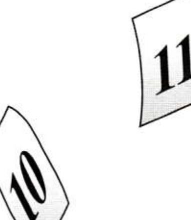
Ausgabe

Die Ausgabe der bearbeiteten Bilder kann direkt vom Programm aus erfolgen. Alle Druckertreiber entsprechen dem GDPS-Standard, dem auch alle anderen TmS-Produkte folgen. Das bedeutet, daß Sie nur einen Treiber für Ihren Drucker benötigen, der von sämtlichen Programmen benutzt wird. Auf monochromen Drukkern können die Bilder als geditherte oder gerasterte Bitmap ausgegeben werden. Auf farbfähigen Druckern können auch Farbausdrucke erzielt werden, die durch-

1989









AKTUELLES

SOFTWARE

BUSINESS

Datenbanken, Tabellenkalkulationen, Handel, FiBu

DFÜ

GRAFIK / CAD

MUSIK

PROGRAMMIER-SPRACHEN

TEXTVERARBEITUNG DTP

SOFTWARELISTINGS PROGRAMMIERPRAXIS

HARDWARE PROJEKTE

HARDWARETESTS

RELAX

Spiele

GRUNDLAGEN

ANWENDUNGEN

ST-ECKE

BÜCHER

aus professionellen Ansprüchen gerecht werden. Selbst auf Nadeldruckern werden akzeptable Ergebnisse erzielt, da die Druckertreiber mit speziellen Algorithmen ein Farbdithering vornehmen, das das letzte aus dem Drucker holt. Tintenstrahldrucker liefern oft bessere Ergebnisse, da die Farben beim Druck auf dem Blatt noch mischbar sind und das Verlaufen der Farben die horizontalen Streifen verhindert, die von Nadeldruckern oft erzeugt werden.

Auch ein professioneller Fotosatz ist möglich. Ab Calamus 1.1 sind frei definierbare Rasterweiten und Winkel möglich, wodurch die Folien für den Vierfarbdruck erzeugt werden können. Grautonbelichtung und Farbbelichtung sind schon heute bei einigen speziellen Belichtungsdiensten möglich. Diese Folien können dann mit bis zu 2540 dpi belichtet werden und geben Ihnen im Druck eine unbertreffliche Qualität.

Zur CeBIT erscheint eine weitere Version des Programms, die für noch speziellere Anwendungen konzipiert worden ist und wiederum einen Meilenstein in der digitalen Bildverarbeitung setzen soll. Man darf gespannt sein. Color Express-Kunden können es im Zuge eines Upgrade-Services erwerben. Mehr über das Programm wollte man uns bei TmS noch nicht verraten.



Die Möglichkeiten bei der Wandlung eines Farbbilds in ein Grautonbild. Es wurden verschiedene Farbanteile bzw. verschiedene Mittelwerte zur Erzeugung des Grautonbilds berücksichtigt.

Preise

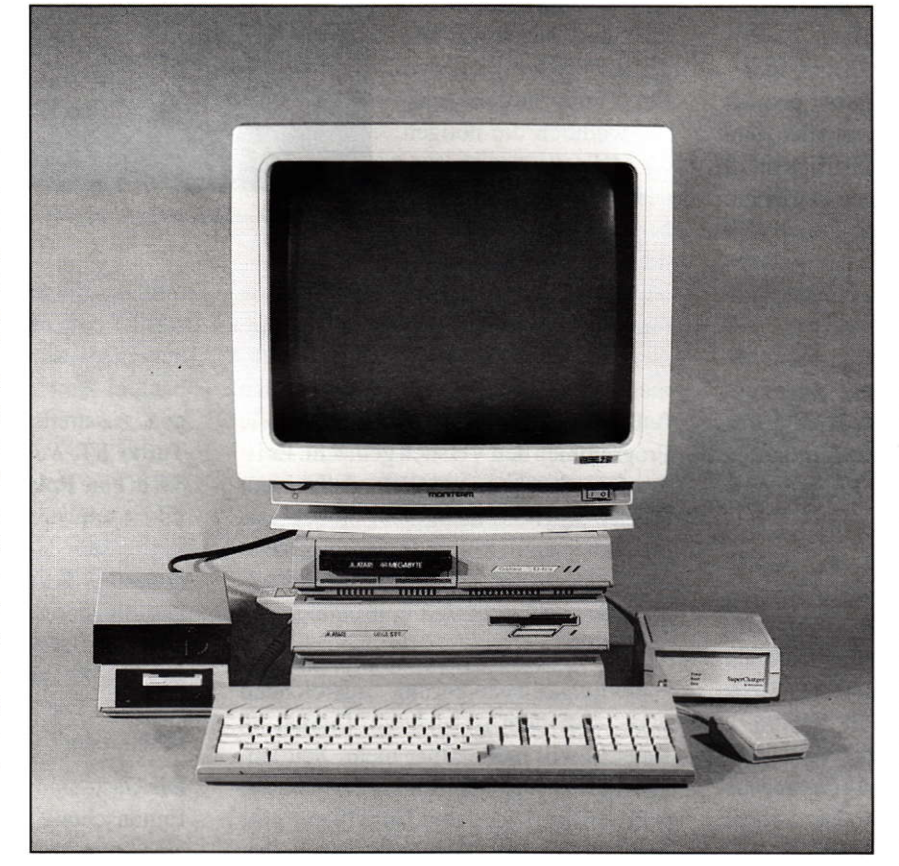
Für das Programm selbst müssen DM 369,- bezahlt werden. Der Löwenanteil des Gesamtpreises fällt auf die Geräte, die zum Einscannen oder zur Ausgabe der Bilder benötigt werden. Ein entsprechender Epson-Scanner (200 dpi, interpoliert bis 400 dpi) kostet beispielsweise DM 4998,-. Allerdings können, wie bereits erwähnt, auch preiswertere Geräte mit ähnlich guten Ergebnissen eingesetzt werden. Der Tintenstrahldrucker, auf dem unsere Bilder ausgedruckt wurden (Canon FP-510) ist für DM 10400,- zu bekommen. Als Grafikkarte wurde bei unseren Beispielen eine MGE von MAXON verwendet (ab 1800,- DM). Für knapp 17000,- DM erhält man also ein Bildverarbeitungssystem, das auch höchsten Ansprüchen gerecht wird.

MP

Bezugsquelle:
TmS GmbH
Cranachweg 4
8400 Regensburg
Tel. (0941) 95163

Traumbild oder wahr?

DER SUPER-ST



Man schreibt das Jahr 1989, Monat November, der Tag ist ein Donnerstag - wie könnte es anders sein? Seit über vier Jahren gehört der ATARI ST zu den beliebtesten Computern in einem Bereich, der schwer zu definieren ist (falls das überhaupt wichtig ist). Wie der menschliche Körper im späten Mittelalter, ist dieser Rechner Grundstein für anatomische Studien, verschiedene Versuche und noch interessantere Erweiterungen.

Alles nur Denkbare, falsch, sogar alles nur Vermutbare wurde auf diesem Rechner ausprobiert. Die Träume vieler engagierter Programmierer wurden Schritt für Schritt Realität. Der Rechner hat zu seiner ursprünglichen Form immer mehr Abstand bekommen. Wir haben der Rolle von Dr. Frankenstein zugestimmt und versucht, uns für Sie eine Hybride, einen Super-ST, auszudenken. Vielleicht haben Sie Spaß daran, an unserem Experiment teilzunehmen.

Unter dem gelben Tuch kann man kaum Formen erkennen. Die Spannung ist unerträglich (es war schon gestern nacht genauso unerträglich, als wäre Vollmond), und die Hände zittern. Viele Augenpaare schauen in der Dunkelheit auf die Gestalt, die sich unter dem gelben Tuch befindet. In den Gedanken vieler Anwesender bewegen sich die Eindrücke des ersten Momentes, als alles noch eine Idee war.

Vielleicht hat alles als kleiner Traum von mehr Leistung angefangen. Eine Art 'Cray II' in Kleinformat. Vielleicht war es purer Neid, als man sich in den Uni-Aulas überall mit irgendwelchen Workstations auseinandersetzen mußte, oder vielleicht war es, wie gesagt, nur die Verwirklichung alter Träume. So alt - älter als der ATARI ST selbst. Man weiß es heute nicht mehr, oder besser gesagt, man will es nicht mehr wissen, weil es nicht wichtig ist. Wichtig ist das neue 'Wesen', das unter dem gelben Tuch 'lebt'. Es wird Zeit, daß diese Spannung zu einem Ende kommt.

Langsam wird die Gestalt von ihrem Schleier befreit. Die ersten Formen lassen sich erblicken, die Spannung läßt langsam nach. Sehr bekannte Formen werden dem Zuschauer immer deutlicher. Ein ATARI ST in seiner MEGA-Version steht da. Eine Festplatte - der Name ist immer noch nicht zu erkennen - leistet ihre Dienste als Massenspeicher. Ein anderes Gerät, das ungefähr die gleichen Maße der Festplatte besitzt, kann man ebenfalls entdecken. Später stellt sich heraus, daß es sich um eine Wechselplatte handelt. Neben diesem Turm steht ein Gerät, das wie ein Floppy-Laufwerk aussieht, aber in seinem Schlitz vorne eine Kassette hat. Es ist ein Band-Streamer. Mehrere Monitore stehen angeblich zur Auswahl. Unter anderem der schon bekannte SM 124, aber auch ein großer 19"-Monitor, der uns mit seinen blinden Augen im Moment nur reflektieren kann. Überall stecken Karten und verschiedene andere Geräte. Das alles ist ziemlich unübersichtlich und fast monströs.

Kein Applaus von seiten der Gäste. Die schauen sich nur fragend in die Augen, ohne ganz zu begreifen, was das alles sein soll. Welches Mysterium kann sich in etwas verstecken, das man schon kennt? Was ist neu an einem MEGA ATARI ST, den man sowieso zu Hause hat? Für die Veranstalter aber ist das Ganze eindeutig. Sie stehen hinter dem Monstrum mit einem Lächeln im Gesicht. Es braucht nur ein kleines 'Klick', und der gewünschte Prometeus ist erweckt. Verschiedene Geräusche und Farben erfüllen den Raum, und auf einmal tritt das Auditorium dem Gerät immer und immer näher.

Der erste Traum wird Wirklichkeit

Der ATARI ST war immer ein sehr schneller Rechner. Als er vor über vier Jahren auf den Markt kam, war seine Geschwindigkeit gegenüber anderen Rechnern, die erhältlich waren, ein absolutes Rennpferd. Die Aufsteiger von Commodore 64 sowie Umsteiger von PC-XT waren (sind immer noch) absolut zufrieden mit der Ausführungsgeschwindigkeit des ATARI STs. Langsam wurden aber die Programme komplizierter, und man hat nach noch mehr Geschwindigkeit verlangt. Die alten Zeiten, in denen man sich mit einem 'Dataset' gequält hat, gerieten rapide in Vergessenheit, und man gab sich mit der 'lächerlichen' 8 MHz-Taktfrequenz nicht mehr zufrieden. Die erste Lösung für diesen Traum war der Einbau eines M68020. Die Pak-Erweiterung brachte mehr Leistung und eine Geschwindigkeitssteigerung mit sich. Da aber das TOS nicht M68020-Code-tauglich war, mußten unbedingt einige Modifikationen im TOS selbst vorgenommen werden. Trotzdem laufen bis heute noch nicht alle Programmen unter dieser Lösung.

Später, viel später, wurde eine andere Richtung eingeschlagen, die plausibler und unkomplizerter war. Die Firma Pro VME baute eine kleine Schaltung, die den ST um einiges pfiffiger machte (siehe ausführlichen Bericht in Heft 7/8 1989). Das war der erste Schritt zur Verwirklichung unseres Traums. Das Hypercache ist schnell eingebaut. Sie müssen Ihren Rechner aufmachen, die eingebaute CPU auslöten (versuchen Sie bitte nicht, sie zu retten - es hat keinen Sinn!) und stattdessen einen Sockel einlöten, der zum Lieferumfang von Hypercache gehört. In diesen Sockel wird die Karte eingesteckt, die den neuen Prozessor und das Cache selbst beinhaltet. Zusätzlich müssen Sie noch zwei Drähte löten: der eine wird mit den Soundchips verbunden und dient dazu, das Cache ein- und auszuschalten, der andere wird auf den Pin 39 des Shifters gelötet, wodurch die nötigen 16 MHz gewonnen werden. Das war's. Jetzt haben wir einen ATARI ST, der mit 16 MHz getaktet ist. Die Steigerung ist gewaltig und funktioniert mit umheimlich vielen Programmen (wir haben es mit vielen Programmen probiert, und bis jetzt hat es mit allen funktioniert. Wir haben aber nicht mit alle Programmen den Versuch gemacht. Es ist insofern durchaus möglich, daß es mit einiger Software nicht geht, obwohl das unwahrscheinlich ist). Einmal eingebaut - und man möchte es nicht mehr missen. Der erste Traum ist fast vollständig.



Bild 2: Ein Farbgroßbildschirm gehört dazu.

Man hätte denken können, daß diese Geschwindigkeitssteigerung ausreichend wäre, aber dem ist nicht so. Einmal im Rausch, will man immer mehr. Seit langem hat man versucht, dieses Problem mit Software zu lösen, und hatte damit gute Erfolge. Mit dem Hypercache wurde aber definitiv eine Barriere durchbrochen. Was könnte man noch tun, um es mit zusätzlichen Mitteln noch schneller zu machen? Mitte letzten Jahres kam ein Programm auf den Markt, das den ATARI ST um einges beschleunigte: eine Art Software-Blitter, der sich mit vielen Programmen verträgt. Er ist ein ACCESSORY, wird einmal eingeschaltet, und das Ergebnis ist sofort auf dem Bildschirm sichtbar (es geht letztlich um diese Ausgabeform). Turbo ST, so heißt dieses Produkt, ist mittlerweile eine Software, die fast jeder besitzt. Allein sie, die mittlerweile in der Version 1.6 vorliegt, schafft eine enorme Steigerung gegenüber einem normalen ST, sogar mit eingebautem Blitter. Aber beide Erweiterungen zusammen, also Hypercache und Turbo ST, verwandeln einen normalen ST in eine Rakete. Da kommt Freude auf. Es ist schwer zu sagen, in welchen Prozenten sich diese Steigerung bewegt. Bei normalen Texten, die über den VDI dargestellt werden, beträgt die Steigerung ungefähr 100%. Werden die Texte über den VT 52 dargestellt, liegt die Steigerung bei fast 150%. Das ist doch ganz schön, oder?

Bild 1: Eine eingebaute MGE-Grafikkarte

Die Gäste sind immer verblüffter. Aus der Enttäuschung der ersten Minuten wird eine immer wachsendere Begeisterung. Buchstaben blitzen über den Bildschirm, Grafiken werden vergrößert und wieder verkleinert in einem Tempo, das man vorher nicht hat ahnen können. Das Monstrum lebt und wird immer größer.

Der zweite Traum: Leistung

Ohne Zweifel ist diese Geschwindigkeitssteigerung durch den Einbau eines Hypercaches und von Turbo ST ein Leistungsgewinn, aber wir möchten jetzt einmal von der Rechenleistung sprechen. Was kann man in dieser Richtung machen? Mit Software ist nicht viel getan. Hier hat ATARI selbst Hilfe geleistet. Es war zwar nicht die erste, aber es war und ist immer noch die günstigste Lösung. Dieser Traum heißt Coprozessor MC 68881. Von ATARI wird er nur für den MEGA ST geliefert, und der Einbau ist

absolut unkompliziert: Der MC 68881 wird einfach in der Busstecker, der sich im Inneren des Rechners befindet, eingesteckt. Andere Hersteller bieten für alle ST-Modelle eine ähnliche Karte an. Da ist der Einbau aber um einiges komplizierter.

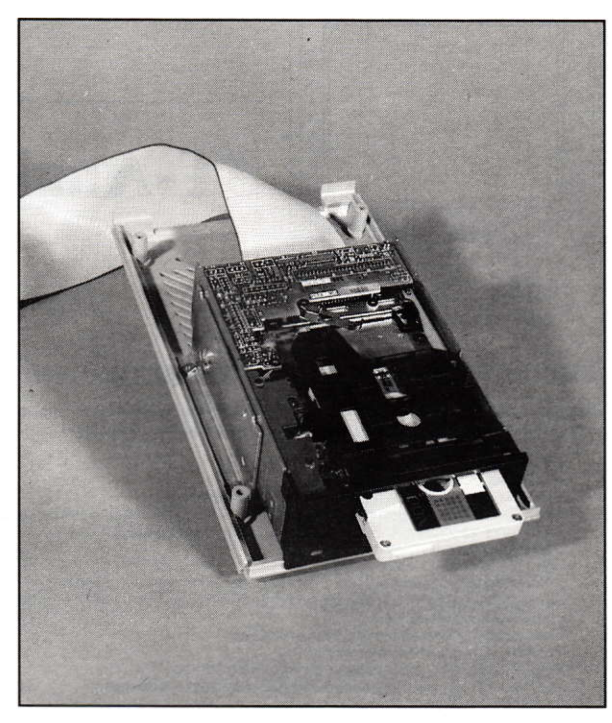
Was bringt diese neue Karte an Leistung? Kann man das wie bei Hypercache oder Turbo ST auf dem Bildschirm sehen? Hier wird es schon schwieriger. Ein Coprozessor bringt seine Leistung im Bereich der Arithmetik, also wenn Sie viel zu rechnen haben und vor allem, wenn diese Rechnungen sehr intensiv und komplex sind. Hierbei muß die Software ebenso mitspielen, denn sonst wird der Coprozessor nicht wahrgenommen. Eine Software, die die Möglichkeiten einer solchen Erweiterung nicht ausnutzt, wird nicht schneller in der Ausführung, egal ob der Coprozessor installiert ist oder nicht. Mittlerweile bieten aber viele Software-Häuser vor allem bei Compiler-Sprachen spezielle Versionen an, die diese sinnvollen Erweiterungen unterstützen.

Das Auditorium wird immer unruhiger und aktiver. Der eine möchte ein spezielles Sieb des Eratosthenes probieren, der andere möchte sehen, wie lange das Monstrum für die vollständige Lösung des Acht-Damen-Problems braucht. Das Gedränge wird immer größer - die Aussteller lächeln im Hintergrund immer weiter.

Der dritte Traum: die Vision

'Ich hatte einen Traum. Ich habe meinen Rechner gesehen. Er hatte einen super-großen Bildschirm auf dem alles sehr scharf und bunt war. Es war wie im Frühling!'

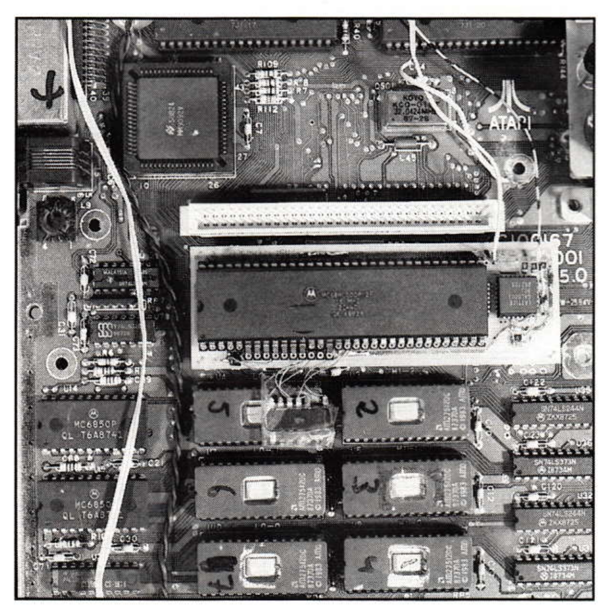
Eines der hervorstechendsten Merkmale des ATARI ST war von Anfang an seine gute SW-Auflösung und seine flimmerfreie Darstellung. Dadurch wurde dem ATARI ST seit seiner Einführung auf dem Markt ein großes Maß an Professionalität zugestanden. Was die Farbe angeht, sieht die Sache absolut anders aus. Die Menge an Farben, die der ST darstellen kann, sowie die geringe Auflösung der unteren Farben macht den ST in diesem Bereich fast unbrauchbar, höchstens für Spiele zu gebrauchen.



Ein Streamer läßt sich z.B. leicht in das Gehäuse einer ATARI-Floppy bauen und über ein SCSI-Interface (Bauanleitung diese Ausgabe) anschließen.

Trotz seiner guten Auflösung im Schwarzweißmodus wuchs das Verlangen nach einer noch höheren Qualität. Als der ATARI ST im DTP-Bereich Fuß gefaßt hatte, wurde noch deutlicher, daß der SW-Monitor einfach zu klein war. Jeder andere Rechner konnte mit einer zusätzlichen Karte einen 19"-Monitor ansteuern, nur der ST nicht. Auch Farben, viele verschiedene sogar, können sehr unterschiedliche Rechner darstellen. Sechs Farben sind auf keinen Fall ein Standard. Man muß mehr können. Viel mehr.

Die MAXON-Grafik-Expansion ist eine Karte, die ebenfalls wie die Coprozessorkarte in den Busstecker des Mega ST eingesteckt wird. Sie hat verschiedene Auflösungen, und das sowohl in Farbe als auch in Monochrom. Bis zu 256 verschiedene Farben auf einer Palette von 262144 kann die MGE gleichzeitig darstellen. Und das bei einer Auflösung von 640x480 Pixeln. Im SW-Modus, besser gesagt im Duochrombetrieb, kann man mit der MGE 1280x960 Bildpunkte darstellen. Die mitgelieferte Software, ein speziell für diese Karte entwickelter VDI-Treiber, ermöglicht das problemlose Arbeiten mit vieler und wichtiger Software, wie zum Beispiel CALAMUS. Neue Horizonte öffnen sich mit dieser Erweiterung sowohl für den Programmierer als auch für den Anwender.



Als 16 MHz-Erweiterung bietet sich Hypercache oder Turbo 16 an.

Die Zuschauer in der zweiten Reihe, die nicht viel von Programmierung verstehen, haben von ihren Ellbogen Gebrauch gemacht und erreichten die vordere Reihe. Fast alle erfahrenen Anwender versuchten, mit der Maus und der Tastatur die Farben oder die Darstellung zu ändern. Der Raum reflektierte ständig die verschiedenen Farben, die das 'Monstrum pausenlos wie ein Chamäleon generierte.

Der vierte Traum: The Great Pretender

Wer erinnert sich heute noch an die ersten Jahre, in denen es nicht viel Software für den ATARI ST gab, und ein CP/M-Emulator eine Brücke zwischen einem Berg bestehender Software und dem ST schuf? Fast niemand, es liegt einfach zu weit in Zeit und Raum zurück. Später, als das Angebot schon sehr groß war, kam der erste gut funktionierende MS-DOS-Emulator: PC ditto. Diese Software-Emulation, die sehr langsam ist, tat fast zwei Jahre ihren Dienst und wurde für viele, die MS-DOS-Software am ST probieren wollten, eine richtige Lösung.

Was aber kann man heute in diesem Bereich machen? Kann mein ST MS-DOS besser emulieren als PC-ditto? Unserer kann es! Es ist ebenfalls wie das Hyperca-

che, die Coprozessorkarte und die MGE eine Lösung, die man im Rechner intern ausführen muß. PC-Speed, so heißt der Emulator, der in unserem Monstrum seine Dienste verrichtet, wird direkt auf die CPU gelötet. Natürlich ist das nicht kinderleicht, aber mit ein wenig Geduld problemlos zu bewältigen. Dann noch ein wenig Software, und unser Super-ST hat sich wieder in einen anderen Rechner verwandelt. Der PC-Speed ist sehr schnell und eine riesige Menge von MS-DOS-Programmen läuft unter dieser Emulation.

Natürlich ist nicht jeder Anwender auf MS-DOS fixiert. Unter uns gibt es auch ein paar Feinschmecker, die, wenn es sich um Emulation handelt, dann eher einen 'richtigen' Rechner emulieren möchten. Zum Beispiel den mit dem Apfel. Sie wissen schon, den 'Mac'.

Beliebt, in Europa aber für Studenten fast unerreichbar, ist der Macintosh von APPLE. Im Grunde ist der ATARI ST für vieles nur eine Art Zwischenlösung. Von dem, was der Geldbeutel nicht schafft, können wir nur träumen und deswegen existiert seit fast zwei Jahren ein Macintosh-Emulator, der im Inneren unseres STs herumgeistert. Man braucht den Rechner nicht aufzumachen, man muß nur einfach einstecken. Dann wieder ein wenig Software (in diesem Fall sogar fremde Software, von Apple nämlich), und man hat auf dem ST einen Macintosh-Rechner.

Jetzt kamen von ganz hinten Herren mit Nadelstreifenanzügen, einem Melonenhut auf dem Kopf und einem Stock in der Hand, betrachteten das 'Mostrum' skeptisch und gingen sofort wieder, als wären sie beleidigt.

Der fünfte Traum: die Masse

Alles fing mit einem kleinen, fast lächerlichen Floppy-Laufwerk an. Es war ziemlich unbrauchbar, weil es von der Kapazität her einfach zu klein war. Die ersten fremden, aber größeren Laufwerke kamen kurz danach. Der Anschluß ist unproblematisch, denn der ST ist von Haus aus schon so vorbereitet, daß man jedes beliebige Laufwerk anschließen kann. Aber der Bedarf nach größeren und schnelleren externen Massenspeichern wurde immer stärker.

Die ersten Festplatten hatten Schwierigkeiten, sie waren langsam, teuer und von der Kapazität her nicht sehr berauschend. Jetzt aber gibt es eine wahre Flut von Platten (siehe in ST 12/89 die Vorstellung verschiedener Platten von Claus Brod). Nicht nur, daß die Platten schneller wurden, auch die Kapazitäten gingen in die Höhe, und die Preise wurden immer niedriger.

Bei unserem Super ST haben wir uns für eine andere Lösung entschieden. Eine etwas komplizierte, aber elegante: Wir haben eine Platte im MEGA ST-Gehäuse installiert. Wie man das macht? Im Grunde ist es relativ einfach, weil genügend Platz und alles, was man an Schnittstellen braucht, vorhanden ist. Wir haben eine Quantum-Platte mit 80 MByte und ein SCSI-Interface von ICD verwendet. Der Host-Aadapter von ICD wird an den ACSI von ATARI angeschlossen, und mit der Software von ICD oder mit den Harddisk-Treibern aus dem Scheibenkleister wird die Platte angesprochen (eine genaue Anleitung, wie man eine Platte mit einem SCSI-Interface einbaut, findet man in dieser Ausgabe der ST Computer. Außerdem wird ein SCSI-Host-Adapter als Hardware-Projekt vorgestellt). Falls Sie persönlich diese Lösung nicht bevorzugen, gibt es im Moment genügend Platten, die man einfach am DMA-Port anschließen kann.

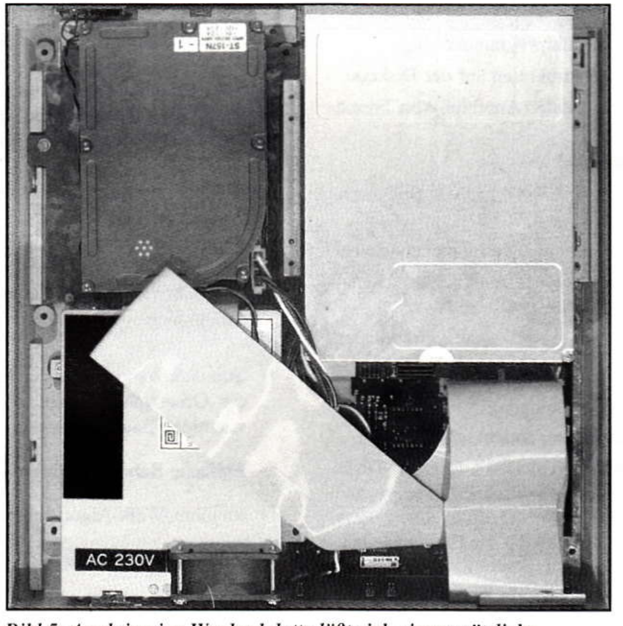


Bild 5: Auch in eine Wechselplatte läßt sich eine zusätzliche Festplatte einbauen (s. diese Ausgabe).

Aber was macht man, wenn die Platte voll ist? Ja, dann brauchen Sie ein Backup davon. Machen Sie das auf Diskette, kann es sehr lange dauern. Bis jetzt war fast nichts anderes möglich. Wir haben uns bei unserem Experiment für die Wechselplatte von ATARI entschieden. Das Megafile 44 stellt ein absolut günstiges Preis/Leistungverhältnis dar. Das Medium sowie die Kapazität stimmen allemal.

Das Auditorium war hellauf begeistert. Funkelnde Augen betrachteten das Monstrum, das sich immer weiter entfaltete, wie ein gutmütiges Ungeheur. Kurz danach leerte sich der Raum. Die Veranstalter, die ihr Lächeln auf dem Gesicht immer noch nicht ausradiert hatten, gingen ebenfalls. Im Raum blieb das 'Monstrum' alleine zurück.

Das ernüchternde Erwachen

Das alles hier ist nicht irgendeine redaktionelle Phantasterei, sondern pure Realität. Der ST, den viele Anwender schon fast abgeschrieben hatten, die schon ungeduldig auf den neuen ATARI-TT warteten, ist durchaus noch erweiterbar. Unsere Träumerei in diesem Artikel ist für viele Anwender eine feste Realität. Ob alles, was wir beschrieben haben, einen vernünftigen Nutzen hat, bleibt jedem selbst überlassen. Ob es unbedingt so teuer sein muß, wie wir es geschrieben haben - mit Sicherheit nicht. Die Alternativen sind gewaltig. Man muß z.B. nicht unbedingt PC-Speed nehmen, wenn man einen MS-DOS-Rechner emulieren möchte, es gibt andere Emulatoren auf dem Markt, wie den Supercharger, den man einfach einstecken kann, ohne den Rechner aufmachen zu müssen. Ebenso gibt es neben Aladin den Spectre 128, oder, wenn man eine höhere Auflösung im SW-Modus erreichen möchte, kann man genauso zwischen den verschiedenen Karten, die die Firma Matrix anbietet, wählen oder direkt bei ATARI einen SM 194 kaufen. Bei der Wahl von Massenspeichern ist die Qual der Wahl noch größer. Man hat eine Reihe von Lösungen, die von uns gar nicht angesprochen wurden, z.B.: Bandstreamer, CD-ROM, RAMs-Bank etc.

Wie Sie sehen, kann man sich diesen Rechner so gestalten, wie man möchte, man muß nur seine eigenen Bedürfnisse kennen und dann anfangen zu suchen.

Schöne Träume!

Martina Pfahl

GANZ SCHÖN DICHT

LAUFWERKS-INFORMATION

Laufwerkskennung: B
Laufwerksname: ________.___
Anzahl der Ordner: 0
Anzahl der Dateien: 0
Bytes belegt: 0
Bytes frei: 1456640

OK

HD-Laufwerke am ST

Wieder fällt eine Barriere, und Sie erleben in dieser "ST-Computer" diese kleine Sensation mit: HD-Laufwerke können mit minimalem Aufwand am ST angeschlossen und betrieben werden. Auf die für ATs und PS/2-Rechner üblichen HD-Disketten passen 1.44 Megabytes (mit HYPERFORMAT sogar einiges mehr), der Datenaustausch mit PCs wird endgültig zum Kinderspiel. Das Ganze präsentieren wir Ihnen in zwei Teilen. Zur Bauanleitung in diesem Heft liefern wir Ihnen in der nächsten Ausgabe gleich ein kleines Formatierprogramm mit, das ein MS-DOS-kompatibles 1.44-MB-Format erzeugt.

Im Laufe der Entwicklung der verschiedensten Diskettenlaufwerke haben sich drei große Aufzeichnungsstandards herausentwickelt:

- *einfache Dichte*: Mit dem FM-Aufzeichnungsverfahren brachte man auf eine doppelseitige, 80spurige Diskette etwa 360 kB unter, 180 kB pro Seite

- *doppelte Dichte*: Das MFM-Aufzeichnungsverfahren brachte eine Verdoppelung der Speicherkapazität auf 720 kB bei einer Datenrate von 250 kBit/s

- *hohe Dichte*: Das MFM-Aufzeichnungsverfahren wird beibehalten, die Datenrate auf 500 kBit/s verdoppelt; die Daten werden also doppelt so dicht auf die Diskette gepackt. Daraus ergeben sich satte 1.44 MB Kapazität, für die man aber spezielle HD-Disketten und HD-Laufwerke braucht.

Was FM und MFM sind, ist in diesem Zusammenhang nicht mal so wichtig; wer mehr wissen will, sei auf [1] verwiesen. Der Floppycontroller WD1772 des ST kann zwischen diesen beiden Aufzeichnungsverfahren hardwaremäßig umgeschaltet werden; im ST ist der WD1772 per Hardware auf das MFM-Verfahren fixiert. Die Datenrate beträgt im MFM-Modus 250 kBit/s. Das bedeutet: man muß sich mit 720 kB, bei Verwendung von HYPERFORMAT oder anderen speziellen Formatierprogrammen, die Lücken auf der Spur effizienter nutzen, auch mal mit 900 kB zufriedengeben. Der Anschluß von HD-Laufwerken blieb dem ST verwehrt, weil der FDC (kurz für *Floppy Disk Controller*) dazu eine Datenrate von 500 kBit/s unterstützen muß - und das tut der WD1772 nicht.

Um diesem Mangel beizukommen, gab es schon viele Ideen:

- einen zum WD1772 halbwegs kompatiblen Floppycontroller einbauen, der auch in den HD-Modus umschaltbar ist. Dafür kommt vor allem der WD2793 in Frage, der eigentlich auch für den TT geplant war. Dummerweise benötigt man zum WD2793 noch relativ umfangreiche Außenbeschaltung, so daß ATARI beim TT wohl darauf verzichtet. Noch ein Grund mehr für uns, die Sache endlich in die eigene Hand zu nehmen.

- andere spekulierten auf komplizierte Host-Adapter/Controller/Laufwerkskombinationen am DMA-Bus, was aber nicht nur teuer käme, sondern auch erhebliche Software-Schwierigkeiten mit sich bringt.

Doch es geht auch viel einfacher. Was bisher unmöglich schien, ist durch die Vorarbeit in [1] und [2] und ein wenig Wagemut Realität geworden: Leicht modifiziert, kann der Floppycontroller des ST HD-Disketten mit einer Kapazität von 1.44 MB formatieren, lesen und beschreiben.

Daraus schließen Sie ganz richtig: Es geht nicht ganz ohne Bastelei ab - Nur-Softies sollten sich jetzt auf die Suche nach einem Lötkolbenartisten in der Bekanntschaft machen. Der Umbau ist zwar einfach und eigentlich unkritisch, aber man weiß ja nie... auf jeden Fall müssen Sie die Grundregeln beherzigen, als da wären unter anderen: Vor dem Öffnen des ST alle Stecker ziehen! Nicht mit Lötzinn die Platine bekleckern! Und: Cool bleiben und Garantieansprüche abschreiben...

Der große Clou

Das Geheimnis ist so trivial, daß viele gleich aufstöhnen werden: "Na, das ist ja... hätte ich mir ja denken können." Die Datenrate, mit der der WD1772 auf die Diskette schreibt, ist abhängig von der Taktrate, mit der er gespeist wird... klingelt's? Normalerweise wird der FDC vom System mit einer Taktfrequenz von 8 MHz versorgt, und Western Digital gibt dazu auch seinen Segen. Daraus errechnet sich sich die MFM-Datenrate von 250 kBit/s. Was aber, wenn man den FDC mit 16 MHz traktierte... da müßte sich doch eigentlich auch die Datenrate auf 500 kBit/s verdoppeln... hmmm... das KANN er doch gar nicht aushalten! Die Dokumentation von Western Digital zum WD1772 sagt: "8 MHz ±0.1%" - aber wen interessiert das? Probieren wir's!

Und tatsächlich schluckt der WD1772 die 16 MHz ohne Murren. Ein wenig wärmer wird er wohl, doch bleibt er allemal unter 40 Grad (gemessen mit einem Wärmestreifen auf der Oberfläche), und das hält selbst die Western-Digital-Dokumentation für eine äußerst behagliche Temperatur.

Der nächste Schritt: Ein HD-Laufwerk, das TEAC-Laufwerk FD235HF, wurde angeschlossen. Dieses Laufwerk hat einige Vorzüge:

- schluckt Disketten doppelter (720 kB) und hoher Dichte (1.44 MB)
- erkennt automatisch, welcher Disk-Typ (doppelte oder hohe Dichte) eingelegt wird, und zeigt dies an einem Pin des Shugart-Busses an
- braucht nur eine Versorgungsspannung (+5V)
- sehr kompakt
- kaum teurer als ein normales 3.5"-Laufwerk

Ran mit dem Laufwerk

Zum Anschluß des TEAC-Laufwerks an den ST müssen auf dem Laufwerk die folgenden Jumper gesteckt sein:

D0 Das Laufwerk reagiert dann auf Signale, wenn das Signal *Drive Select 0* auf dem Shugart-Bus aktiviert wird, fühlt sich also als Laufwerk 0 am Shugart-Bus.

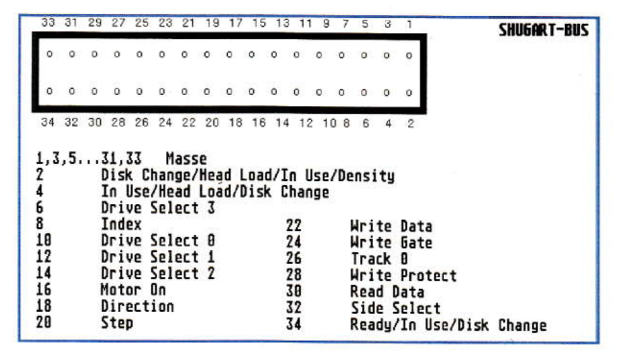
OP automatische Erkennung des Diskettentyps im Laufwerk

HHO Laufwerk liefert HD-OUT-Signal an Pin 2 (High = High-Density-Diskette, Low = Double-Density)

Bei den verwendeten TEAC-Laufwerken fehlte dazu jeweils ein Jumper; man besorge sich also rechtzeitig ein solches Dingelchen, was sich sowieso empfiehlt, weil sich das Zeug gerne in irgendwelche Ecken verflüchtigt, an die man nie mehr herankommt. Man kann auch zwei Einzel-Pins von gedrehten Kontakten aufstecken und miteinander verlöten.

Der Anschluß von Floppy-Laufwerken ist Gott sei Dank standardisiert: Der sogenannte *Shugart-Bus* hat sich durchgesetzt. Dieser 34polige Anschluß, dessen Belegung in Bild 1 dokumentiert ist, bietet alle notwendigen Signale zur Steuerung von Floppies. Maximal vier Laufwerke können per Shugart-Bus angeschlossen werden, die über die vier Drive-Select-Leitungen ausgewählt werden. Die Nummer, unter der sich jedes Laufwerk angesprochen fühlt, kann im Laufwerk an den DS-Jumpern konfiguriert werden - siehe oben.

Der Floppy-Anschluß am ST benutzt nur die wichtigsten Signale des Shugart-Busses (siehe Bild 2). Exotischere Signale wie etwa "High Density" vom Pin 2 des Shugart-Busses sind



Pin	Signal	Pin	Signal
1,3,5...31,33	Masse		
2	Disk Change/Head Load/In Use/Density		
4	In Use/Head Load/Disk Change		
6	Drive Select 3		
8	Index	22	Write Data
10	Drive Select 0	24	Write Gate
12	Drive Select 1	26	Track 0
14	Drive Select 2	28	Write Protect
16	Motor On	30	Read Data
18	Direction	32	Side Select
20	Step	34	Ready/In Use/Disk Change

Bild 1: Pin-Belegung beim Shugart-Bus

ausgespart. Das ist ärgerlich, weil wir gerade dieses Signal später im Rechner brauchen werden, um automatisch zwischen 8 und 16 MHz umschalten zu können.

Dem Bild 2 kann man auch entnehmen, welche Signale des Shugart-Busses auf den Floppyport des ST gegeben werden müssen. Alle ungeraden Pins des Shugart-Busses verbindet man miteinander und verlötet sie mit Pin 3 und Pin 7 (Masse) des Floppyports am ST. Viele Firmen liefern Adapterkabel vom Shugart-Bus zum Floppyport des ST als Zubehör; wir haben uns zum Beispiel Kabel von FSE Computersysteme in Kaiserslautern besorgt.

Zwei Kleinigkeiten, die schon vielen das Leben schwer machten, die Zweitlaufwerke anschließen wollten:

- Die Signale am Shugart-Bus werden in den Laufwerken über Pull-up-Widerstände auf +5V gelegt. Nur in einem der angeschlossenen Laufwerke dürfen diese Pull-up-Widerstände aber installiert bleiben, um die Ausgänge des Rechners nicht zu überlasten. Der ST ist hier besonders empfindlich, da insbesondere die

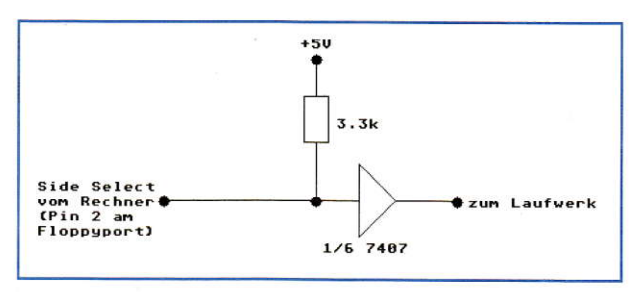
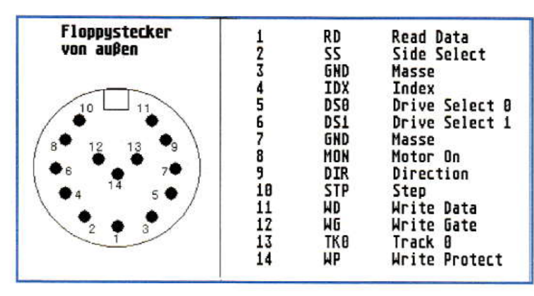


Bild 3: Pufferung des Side-Select-Signals

drei Selektionsleitungen *Drive Select 0, Drive Select 1, Side Select* im Rechner ungepuffert sind. Diese Signale werden im ST vom Soundchip erzeugt. Können aus irgendwelchen Gründen die Pull-up-Widerstände im Zweitlaufwerk nicht entfernt werden (und beim TEAC FD235HF geht das wirklich nicht), sollte man das Side-Select-Signal rechnerintern puffern - siehe Bild 3.

- Bei 1040ern und MegaSTs ist bereits ein Laufwerk eingebaut. Am Pin DS0 des Floppyports liegt hier nicht etwa das Selektionssignal für das eingebaute Laufwerk an, sondern das für das externe Laufwerk. Der Pin DS1 ist unbelegt. Durch dieses Kreuzen der Leitungen erreicht man, daß alle angeschlossenen Laufwerke als Shugart-Laufwerk 0 gejumpert werden können (siehe oben).

Daß das TEAC FD235HF mit nur einer Versorgungsspannung von +5V auskommt (Pin-Belegung des Stromversorgungssteckers siehe Bild 4), macht die Sache leichter. Besitzer eines MegaSTs finden beispielsweise in der Nähe des Netzteils einen vierpoligen Konnektor, an dem sich ein Zweitlaufwerk Saft besorgen kann. Die Netzteilkapazität reicht dazu aus. Beim 1040 sowie bei 520ST und 260 ST wird man allerdings auf ein externes Netzteil ausweichen.



Pin	Kürzel	Signal
1	RD	Read Data
2	SS	Side Select
3	GND	Masse
4	IDX	Index
5	DS0	Drive Select 0
6	DS1	Drive Select 1
7	GND	Masse
8	MON	Motor On
9	DIR	Direction
10	STP	Step
11	WD	Write Data
12	WG	Write Gate
13	TK0	Track 0
14	WP	Write Protect

Bild 2: Floppy-Port des ST

Im Prinzip funktioniert der beschriebene Umbau mit jedem *Multifunktionslaufwerk (MF-Laufwerk)*, das ähnliche Eigenschaften wie das TEAC FD235HF bietet. Weitere Beispiele für solche Laufwerke: TEAC FD55GFR (5.25", wahlweise 720 kB oder 1.2 MB, 40 oder 80 Tracks), Chinon FZ-506 (ebenfalls ein 5.25"-Laufwerk) und NEC1137 (3.5"-Laufwerk). Getestet haben wir bisher die beiden TEAC-Laufwerke. Zu anderen Laufwerken sollten Sie sich ein Handbuch besorgen, um die Position und Funktion bestimmter Jumper eruieren zu können. Mehr Tips zum Anschluß von Zweitlaufwerken finden Sie in [1].

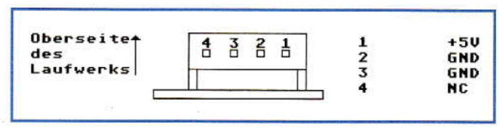


Bild 4: Spannungsversorgung beim TEAC FD235HF

Hertzschmerzen

Zuallererst die Frage: Woher holen wir denn nun die 16 MHz? Natürlich kann man dazu einen externen Quarzoszillator einsetzen. Aber andererseits liegen im ST die 16 MHz am Shifter bereit, wo man sie am Pin 39 leicht

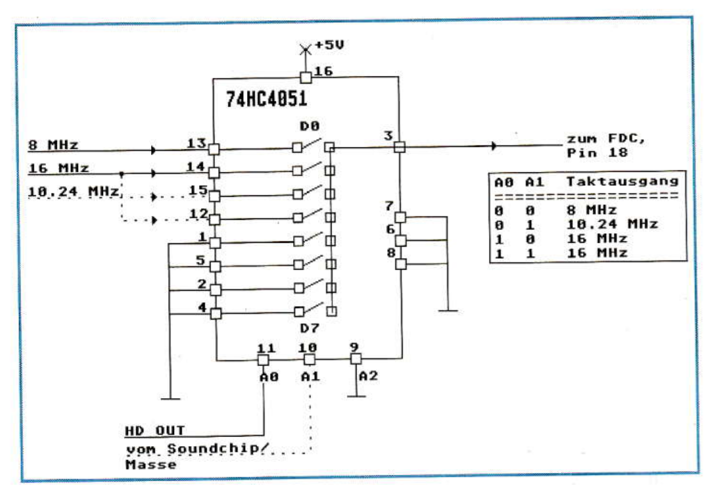


Bild 5: Pin-Belegung beim WD1772 - wichtig ist CLK!

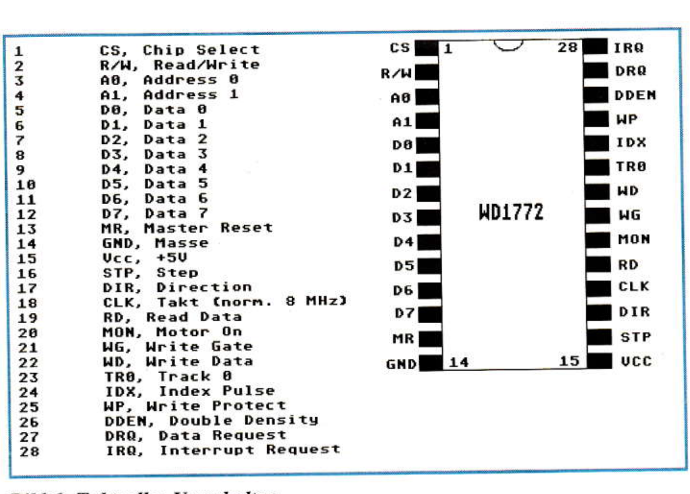


Bild 6: Taktvolles Umschalten

anzapfen kann. Dort lötet man eine Litze an, die natürlich lang genug sein muß, um den Weg bis zum Floppycontroller zu überbrücken. Dieser Weg ist in den verschiedenen ST-Rechnern unterschiedlich lang, nehmen Sie also vorher Maß. Die Litze sollte aber auch so kurz wie irgend möglich sein, denn es geht hier um ein störempfindliches hochfrequentes Signal.

HYPERCACHE von proVME zapft ebenfalls das 16-MHz-Signal am Shifter an. Da der Shifter an diesem Pin nur leicht belastet werden sollte, empfehlen wir für diesen Fall einen separaten Quarzsozillator. In einem älteren 260ST brach beim Anschluß der separaten Taktumschaltung - auch ohne HYPERCACHE - der Shifter zusammen, weswegen wir auch in diesem Falle einen externen Takt einspeisen mußten. Bei allen anderen umgerüsteten Rechnern reichte das Shifter-Signal völlig aus (mehrere Megas und andere STs). Seien Sie trotzdem auf alle Fälle zärtlich zu Ihrem Shifter, er ist ein sensibles Wesen.

Nun haben wir ein 16-MHz-Signal, das wir immer dann auf den FDC geben wollen, wenn eine High-Density-Diskette im TEAC-Laufwerk liegt und selbiges selektiert ist. Ansonsten wollen wir weiter 8 MHz verwenden, um auch mit normalen Disketten weiterarbeiten zu können. Diese 8 MHz liegen am Pin 18 des Floppycontrollers an. Um die 8 MHz umschalten zu können, bieten sich zwei Lösungen an:

- Pin 18 am FDC in der Mitte abkneifen; das 8-MHz-Signal wird dann von der unteren Hälfte des Pins zur Umschaltplatine geführt, das gewünschte Taktsignal, das von der Umschaltplatine kommt, wird auf die obere Häfte des Pins gegeben. Aufpassen, daß von beiden Enden des Pins genug Angriffsfläche übrigbleibt!

- FDC auslöten und sockeln, Pin 18 wegbiegen

Die letztere Lösung ist natürlich die bessere, auch für den Fall, daß irgendwann einmal im Leben Ihres ST der Floppycontroller Schaden

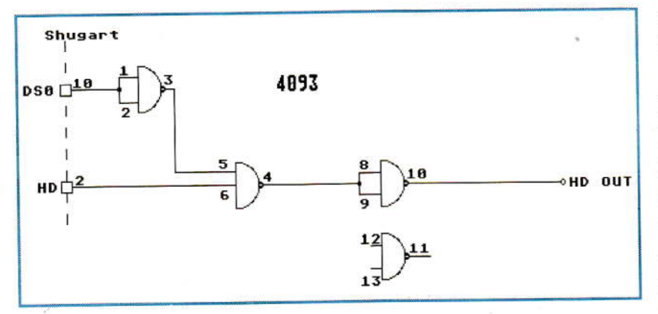


Bild 7: Das HD-Signal vom Laufwerk wird angepaßt

nehmen sollte. Das Auslöten des gesamten Floppycontrollers ist andererseits aber auch nicht einfach.

Zur Umschaltplatine (Bild 6): Wir verwenden für diesen Zweck den Multiplex-Baustein 74HC4051, der abhängig vom Zustand dreier Eingangsleitungen einen von acht weiteren Eingängen auf den Ausgang legt. Natürlich würde es für unseren Zweck auch ein einfacherer Baustein wie der in [2] verwendete 74LS157 tun, aber wir haben mit Bedacht (!) auf Erweiterbarkeit Wert gelegt. So kann man also - wie in Bild 6 gestrichelt dargestellt - die Schaltung leicht abändern, um auch noch einen dritten Takt (10.24 MHz wie in [2] verwendet) einzuspeisen. Auf diese Weise kann man auf ein und demselben Laufwerk wahlweise mit 8, 10.24 oder 16 MHz arbeiten (doppelte Dichte, HYPER DENSITY, hohe Dichte).

Die Umschaltung zwischen 8 und 10.24 MHz geschieht - wie schon in [2] vorgeschlagen - über Pin 15 des Soundchips (Bit 6 im Port A) und damit beispielsweise über das Accessory HYDSEL.ACC von Jürgen Stessun (ST-Computer 7/8 '89). Zwischen 8 und 16 MHz schaltet man am Pin 11 (A0) des 74HC4051 um (High = 16 MHz). Dazu braucht man ein Signal vom Laufwerk, das anzeigt, welcher Diskettentyp eingelegt wurde. HD-Disketten haben eine zusätzliche Kerbe gegenüber der Schreibschutzöffnung, um ihren Typ bekanntzugeben. Das TEAC-Laufwerk erkennt das über eine Lichtschranke und meldet es - wenn man es entsprechend über Jumper einstellt, siehe oben - am Pin 2 des Shugart-Bus-Anschlusses.

Dabei gibt es allerdings einen Haken: Das TEAC-Laufwerk meldet auf diesem Pin "High" (+5V), wenn es nicht selektiert ist. Ein "High" zeigt aber - wenn das Laufwerk selektiert wird - auch an, daß eine HD-Diskette eingelegt ist. Das ist dann fatal, wenn man mehr als ein Laufwerk am ST betreibt. Nehmen wir an, wir haben ein normales Laufwerk als A und das TEAC-Laufwerk als B ange-

```
 1:  R$=Chr$(27)+"p"
 2:  O$=Chr$(27)+"q"
 3:  Print At(28,1);R$;"Steprateneinstellung";O$
 4:  Print At(28,2);R$;"    by Claus Brod     ";O$
 5:  Print At(28,3);"Aus SCHEIBENKLEISTER, dem"
 6:  Print At(28,4);"etwas anderen Floppybuch"
 7:  Print
 8:  Fehler%=Gemdos(32,L:0)
 9:  ' Ab in den Supervisormodus (nicht mehr abbrechen, bitte!)
10:  Dpoke &H440,0        ! 0=6ms, 1=12ms, 2=2ms, 3=3ms
11:  ' In Systemvariable schreiben
12:  P%=Lpeek(&H46A)
13:  Call P%
14:  ' hdv_init aufrufen
15:  '
16:  Fehler%=Gemdos(32,L:Fehler%)
17:  ' zurück in den Usermodus
18:  '
19:  Void Bios(7,0)
20:  ' Nachlaufen beenden
```

Listing 1: Steprateneinstellung in GFA-BASIC

```
 1:  /* newstep: Einstellen der Steprate (für beide Laufwerke)
 2:  ** Written 1988 by Claus Brod
 3:  ** Original in GFABASIC, siehe SCHEIBENKLEISTER
 4:  */
 5:
 6:  #define  seekrate                        0x440L
 7:  #define   _hdv_init                      0x46AL
 8:
 9:  #include <stdio.h>
10:  #include <tos.h>
11:  #include <stdlib.h>
12:
13:  void step(int);
14:  void main(int,char **);
15:  void dpoke(unsigned long, int);
16:
17:  /* dpoke: Schreibt inhalt.w in adresse */
18:  void dpoke( adresse, inhalt )
19:  unsigned long adresse;
20:  int inhalt;
21:  {
22:    *(int *)adresse = inhalt;
23:  }
24:
25:
26:  /* step: Steprate einstellen */
27:  void step(steprate)
28:  int steprate;
29:  {
30:    long hdv_init, old_stack;
31:
32:    old_stack = Super(0L);
33:    dpoke(seekrate, steprate);
34:    hdv_init = *(unsigned long *)_hdv_init;
35:    (void) Super( (void *)old_stack );
36:
37:    Supexec(hdv_init);
38:  }
39:
40:  void main(argc, argv)
41:  int argc;
42:  char *argv[];
43:  {
44:    char newstep = '0';
45:
46:    puts("Stepraten einstellen - (C)1989 Claus Brod");
47:
48:    if (argc > 1)
49:    {
50:            newstep = *argv[1];
51:
52:            if ((argc != 2) || (newstep < '0') ||  (newstep >'3'))
53:            {
54:                    puts("USAGE: newstep [0|1|2|3]");
55:                    puts("        0=6ms, 1=12ms, 2=2ms, 3=3ms");
56:                    Cnecin();
57:                    exit(1);
58:            }
59:    }
60:    step((int)(newstep - '0'));
61:    exit(0);
62:  }
```

Listing 2: Steprateneinstellung in Turbo C

schlossen. Nun greifen wir auf A zu; B ist damit deselektiert und liefert ein “High” auf Pin 2. Unsere Umschaltung, der dieses Signal zugeführt wird, würde das irrtümlich so interpretieren, als wäre eine HD-Diskette im gerade selektierten Laufwerk A - der FDC schaltet auf 500 kBit/s um, und schon geht alles schief. Wir brauchen also eine kleine Modifikation des HD-Signals vom TEAC-Laufwerk, das diesen Effekt vermeidet (Bild 7). Dazu genügt beispielsweise ein einfaches IC4093, dessen NAND-Gatter uns das gewünschte HD-OUT-Signal liefern: “High”, wenn das Laufwerk selektiert und eine HD-Diskette eingelegt ist, sonst “Low”.

Zusammengefaßt funktioniert das alles so: Das selektierte Laufwerk meldet dem Rechner über Pin 2 am Shugart-Bus, welcher Diskettentyp eingelegt ist. Abhängig davon wird entweder ein 8- oder ein 16-MHz-Takt auf den FDC gegeben.

Für den Pin 2 am Shugart-Bus fehlt dem ST an seinem Floppyport das entsprechende Pendant - man muß sich hier also mit einem Extrakabel behelfen, das man durch irgendeine Öffnung des ST zur dort eingebauten Taktumschaltung hineinführt.

Die Taktumschaltung, die man sinnvollerweise im Rechner in der Nähe des Floppycontrollers installiert und befestigt, läßt sich ebenso wie die Modifikation des HD-OUT-Signals noch leicht auf einer Lochrasterplatine aufbauen. Am günstigsten ist es, den FDC zu sockeln und in den Sockel dann eine Platine zu stecken, die die Schaltung und einen weiteren Sockel für den FDC beinhaltet. Eine solche Lösung ist in Arbeit.

Das TEAC FD55GFR (5.25"-HD-Laufwerk) liefert am Pin 2 des Shugart-Busses leider kein HD-Signal; allerdings kann man das Laufwerk immerhin von außen über diesen Pin auf den gewünschten Modus einstellen. Hier legt man am besten das Signal zwischen Pin 2 des Shugart-Busses und dem ersten daran angeschlossenen NAND-Gatter über einen Schalter auf +5V (Schalter geschlossen: High Density, Schalter offen: Double Density).

Schritt für Schritt

Legt man nun eine HD-Diskette ins Laufwerk und selektiert das Laufwerk, schaltet der FDC auf 16 MHz - alles läuft doppelt so schnell. Das betrifft neben der eigentlichen Datenübertragung auch die Step-Impulse, die nun in noch kürzeren Abständen auf das Laufwerk einprasseln! Bei der voreingestellten Step-Rate von 3 ms der ATARI-Laufwerke verkürzt sich die reale Step-Rate damit auf 1.5 ms - viel zuwenig auch für die schnellsten Laufwerke. Man kann den WD1772 aber auch anweisen, alle Kopfbewegungen mit einer Step-Rate von 6 ms auszuführen - so ergibt sich wieder eine reale Step-Rate von 3 ms, und alles ist in schönster Ordnung.

Glücklicherweise existiert unter TOS eine Systemvariable (bei $440), in der die aktuelle Step-Rate des Systems steht. Es reicht zwar nicht, diese einfach nur neu zu setzen, aber mit einer in [1] erstmals veröffentlichten Methode gelingt es, den FDC von TOS aus ohne direkte Controller-Programmierung auf eine andere Step-Rate umzustellen. Diesen Trick wenden die beiden Programme in Listing 1 und Listing 2 an; das erstere zeigt eine Ausprägung in GFA-BASIC, die nur die Step-Rate auf 6 ms stellt und sich sofort wieder verabschiedet - ideal also für den Autoordner. Listing 2 zeigt, wie man so etwas in TURBO C macht; hier kann man auch optional in der Kommandozeile einen Code für die neue Step-Rate (0 = 6 ms, 1 = 12 ms, 2 = 2 ms, 3 = 3 ms) übergeben.

Bevor man also auf ein HD-Laufwerk erstmalig zugreift, sollte man eines der beiden abgedruckten Programme gestartet haben, damit sich der Lesekopf nicht verstolpert. Eine Anmerkung noch zu den beiden Step-Raten-Manipulatoren: Sie stellen die neue Step-Rate für BEIDE Floppy-Laufwerke ein, was an der Methode liegt. Andere Programme, beispielsweise aus der PD-Sammlung oder auf der Diskette zu [1], können die Step-Rate auch getrennt für einzelne Laufwerke einstellen.

Wollen Sie vom HD-Laufwerk und einer HD-Diskette booten, muß auf dieser Systemdisket-

te als allererstes der Autoordner angelegt und darin als erstes Programm ein Step-Raten-Umschalter kopiert werden. So erreicht man, daß das Step-Raten-Programm komplett auf Track 0 zu liegen kommt. Beim Laden dieses Programms braucht das Laufwerk den Kopf also nicht zu bewegen - die allzu hektischen Step-Impulse des FDC (3 ms / 2) stören nicht weiter.

Nicht nur der *Abstand* zwischen den einzelnen Step-Impulsen, sondern auch deren *Impulslänge* verkürzt sich bei der hohen Taktfrequenz. Es mag Laufwerke geben, die damit nicht zurechtkommen. Die getesteten TEAC-Laufwerke akzeptierten auch die etwas überhasteten Step-Impulse ohne Widerwort; sollten Sie ein Laufwerk haben, das nicht so tolerant ist, werden Sie ihm eventuell mit einem Mono-Flop in der STEP-Leitung (Pin 10 des Floppyports) nachhelfen müssen.

Damit sind wir am Ende des ersten Teils angekommen. Wie versprochen, gibt es in der nächsten Ausgabe u.a eine Formatier-Routine für unsere HD-Laufwerke. Falls große Nachfrage ist, werden wir versuchen, zusätzlich zum Artikel fertige HD-Laufwerke für die reinen Anwender des ST käuflich anzubieten. Haben Sie also Interesse, schreiben Sie an:

MAXON Computer GmbH
Stichwort: HD-Laufwerk
Industriestr. 26
D-6236 Eschborn

Joachim Bohs / CB / Anton Stepper

Literatur:

[1] Brod, Stepper: SCHEIBENKLEISTER II, MAXON, Eschborn 1989, ISBN 3-927065-00-5

[2] Jürgen Stessun: "Dem Floppycontroller Dampf gemacht", ST-Computer 7-8/89

[3] Jankowski, Rabich, Reschke: ATARI ST Profibuch, Sybex 1988, ISBN 3-88745-563-0

[4] C-Referenzbuch, Sybex 1987, Olaf Hartwig, ISBN 3-88745-503-7

[5] Erfolgreich programmieren in C, J. A. Illig, Sybex 1987 (4), ISBN 3-88745-055-8

[6] Handbuch zu Laser-C

[7] Handbuch zu Turbo-C

[8] Western Digital: Storage Management Products Handbook 1986

Man geht nicht ohne

STACY



Die Lust auf einen Laptop hatte den Autor schon lange gepackt. Und jetzt steht er vor mir, und jedes der Worte, das Sie hier lesen, wandert über seine Tasten. Aber er ist auch wirklich hübsch. Anthrazitfarben, geradezu edel, mit sanften Rundungen.

Dennoch, er kann nicht verleugnen, wes Geistes Kind er ist. ATARI konnte nicht aus seiner Haut und hinterließ, wenn Sie mir die Übertreibung nachsehen wollen, sozusagen ein kleines Kainsmal auf seiner Stirn.

Ausstattungs-Zweifalt

Doch nun, ganz prosaisch, ans Werk. Geben wir Ihnen etwas Informativeres zu lesen: STACY gibt es zur Zeit in zwei Ausführungen (andere sind noch nicht lieferbar, aber geplant): ganz klein und ganz groß. Wir haben, als privilegierte Klasse der meinungsbildenden Wirtschaftszweige, natürlich eine groooooße Version, mit allem Drum und Dran. In Zusammenhang mit STACY heißt das: 4 volle Megabyte Speicher, ein doppelseitiges Diskettenlaufwerk (verträgt weder FASTLOAD noch 11-Sektor-Disketten; bei Spur 80 war Schluß) und eine 40 Megabyte-Harddisk. Der kleine Kollege hat ein Megabyte Speicher und gar keine Harddisk. Da der Speicher aber in Form von SIMM-Modulen nachgerüstet werden kann, wird man sicher bald entsprechende Erweiterungen, auch mit Platte, kaufen können. Von wegen kaufen: Rechnen Sie in den Kaufpreis ein paar dieser aparten Handschellen ein, mit denen in Gangsterfilmen die Geldkoffer an ihren Trägern befestigt werden. Der kleine Computerkoffer ist nämlich nicht ganz billig: 6498,- DM kostet die große Ausführung (in Worten sechstausendvierhundertachtundneunzig) und der Kleine kostet auch noch 3698,- Mark.

Eigentlich ist STACY ein ganz normaler Mega-ST. Von den normalen Serienmodellen unterscheidet er sich, was die inneren Werte betrifft, im wesentlichen nur durch das modernere TOS 1.4. Naja, das dient sicher der Identitäts- und Klassenbildung der ATARI-Benutzer (Normal- und Luxusausführung): "Was, Du mußtest Dein TOS 1.4 dazukaufen? Also, bei mir ist das selbstverständlich im Preis drin...". Aber mit diesen häßlichen grauen Schuhkartons will sich ja sowieso niemand mehr sehen lassen. Übrigens wird STACY's Uhr ebenfalls gepuffert.

Klappenvielfalt

Um die Spannung bei unseren Lesern noch ein wenig zu erhalten, werde ich jetzt erst noch ein wenig über Schnittstellen und ähnliche Kleinigkeiten plaudern, bevor ich auf die eigentlich interessanten Fragen zu sprechen komme. Sämtliche Schnittstellen des Mega-ST sind ordentlich herausgeführt, sogar die für den normalen Monitor, der parallel zum LC-Display betrieben werden kann. Auch der Reset-Taster und der ROM-Port sind vorhanden, ebenso zusätzliche Anschlüsse für Maus und Joystick. (Hier sei schon einmal verraten, daß STACY einen eingebauten Trackball besitzt, der normalerweise die Maus ersetzt.)

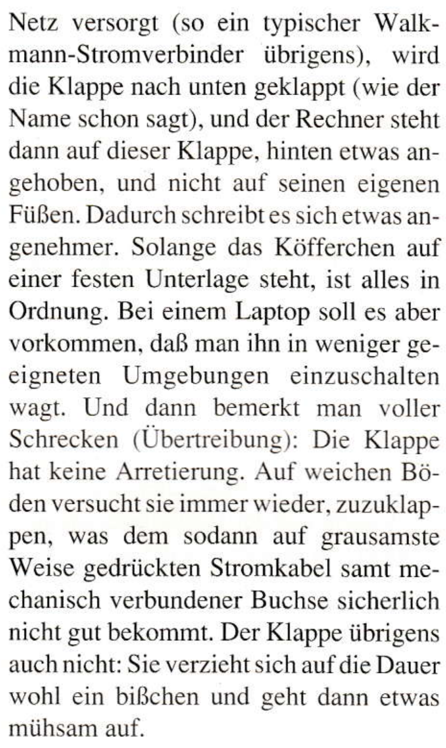
Für Nostalgiker ist der Netzanschluß gestaltet: Das mitgelieferte externe Netzteil ist eindeutig ein dem originalen, dem echten und ursprünglichen 520 ST-Netzteil verwandtes Exemplar. Zeitgemäßer, kleiner und leichter natürlich, aber doch verwandt, kein Zweifel. Alle Buchsen verstecken sich, edles understatement, hinter Klappen, um das Design nicht zu stören. Es gibt sogar eine Klappe, hinter der sich der Mega-ST-Systembus versteckt. Aber offensichtlich schämt er sich in dieser Umgebung seines technokratischen Designs; bei unserem STACY läßt sich die Klappe ohne Säge beim besten Willen nicht öffnen. Laut Mitteilung von

ATARI handelt es sich dabei aber um ein Versehen; normalerweise sollte sie aufgehen.

Übrigens kann man kleine STACYs um Speicher erweitern, ohne das Gerät auseinanderzunehmen: Die Module befinden sich unter einer kleinen Klappe auf der Unterseite des Computers, die man mit nur einer Schraube öffnen kann. Gleiches (man höre und staune) gilt auch für die Betriebssystem-ROMs (eigene Klappe). Der Grund für diese unerwartet praktische Lösung ist, daß das Gehäuse nicht ganz einfach (die Schrauben sind gut versteckt, und ohne Spezialwerkzeug wird es schwierig) zu öffnen ist.

So komm' nun Strom und fließe

Der Netzschalter hat sich leider verlaufen. Oder die Batterien-Industrie hat so viele FAX-Messages an ATARI geschickt, bis das Entwicklungsteam kapituliert hat. Jedenfalls ist dieser Kippschalter, der doch über Sein oder Nichtsein Ihrer Batterien bestimmt, mit höchster Präzision an einer der wenigen Stellen montiert, an der man dem Rechner, wenn man ihn denn so mit sich herumträgt, garantiert ständig versehentlich Leben einhaucht. Dem strengen Liberalismus verhaftet, verweigert STACY auch bei geschlossener Klappe nicht seinen Dienst (ein Schmetterling paßt bestimmt noch zwischen Tastatur und Display, aber er kriegt die Tasten nicht runter), Warnungen sind auch unter der Würde dieses Computers (Ein kleiner 'Pieps' zum Einschalten?). Ein kleiner Kontakt, der den geschlossenen Computer ausschaltet, wäre ganz praktisch.



Wenn wir schon gerade bei den Konstruktions-Genialitäten sind: Ganz besonders ist die große Klappe auf der Rückseite gelungen, hinter der sich die meisten Anschlüsse verbergen: Wenn man den Rechner via Netz versorgt (so ein typischer Walkmann-Stromverbinder übrigens), wird die Klappe nach unten geklappt (wie der Name schon sagt), und der Rechner steht dann auf dieser Klappe, hinten etwas angehoben, und nicht auf seinen eigenen Füßen. Dadurch schreibt es sich etwas angenehmer. Solange das Köfferchen auf einer festen Unterlage steht, ist alles in Ordnung. Bei einem Laptop soll es aber vorkommen, daß man ihn in weniger geeigneten Umgebungen einzuschalten wagt. Und dann bemerkt man voller Schrecken (Übertreibung): Die Klappe hat keine Arretierung. Auf weichen Böden versucht sie immer wieder, zuzuklappen, was dem sodann auf grausamste Weise gedrückten Stromkabel samt mechanisch verbundener Buchse sicherlich nicht gut bekommt. Der Klappe übrigens auch nicht: Sie verzieht sich auf die Dauer wohl ein bißchen und geht dann etwas mühsam auf.

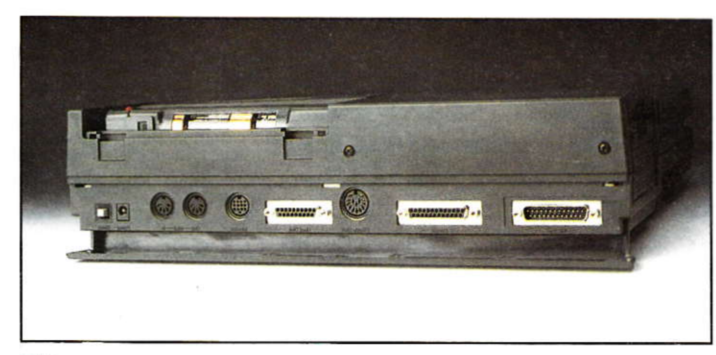


Bild 1: Die Kehrseite von STACY mit allen gewohnten Schnittstellen und rechts oben dem Batteriefach.

Langsam tasten wir uns vor, dorthin, wo die Wildnis beginnt: Wenn man das Display zuklappt, kann man für einen kurzen Moment, in einem ganz bestimmten Winkel nur, ein kleines Glitzern sehen, und das ist er: der Batteriekasten. Gut erreichbar, hinten und auf der Oberseite gelegen. Darinnen findet sich, so man die Klappe (schon wieder eine, aber problemlos) öffnet, einen Batteriehalter für zwölf Babyzellen, die Ihrem Computer alles geben, was er (außer Ihnen selbstverständlich) braucht. Eine kleine Weile natürlich nur. Beim großen STACY halten die Batterien bei normalem Betrieb ungefähr anderthalb Stunden, aber das ist abhängig von Harddisk-Betrieb (so selten wie möglich), Display-Einstellung (siehe unten) etc. Der Kleine kommt bestimmt etwas länger ohne Infusionen aus dem Netz aus. STACY hätte eine stromsparende Harddisk- bzw. Displayabschaltung bei längerer Benutzerpause sicherlich gut gestanden. Übrigens, ATARI liefert einen Satz Batterien für den Einsteiger mit.

Um Spekulationen vorzubeugen: Mit dem mitgelieferten Netzteil kann man keine Akkus aufladen. Sie haben einen ATARI vor sich. Es paßt auch nicht, wenn man die Batterien herausnimmt, in den Batteriekasten, obwohl es kaum größer ist. Sie haben einen ATARI vor sich. Sowas wäre ja praktisch. Außerdem liefert das Netzteil 16.5 V, so daß man die üblichen Lade-/Netzgerätkombinationen auch nicht verwenden kann. Unterwegs wird man Sie also mit einem zweiten Koffer, entweder voller Batterien oder vollgestopft mit Netzteilen und Ladegerät, antreffen. Warum ein Gerät in dieser Preisklasse nicht mit einem Ladegerät ausgestattet ist, wird mir wohl ein Rätsel bleiben. ATARI zuckt mit den Achseln (auf Anfrage).

STACY warnt Sie übrigens nicht, wenn die Batterien zu schwach zum Speichern werden. Ihre Daten sterben einen schnellen Tod. Ich hoffe, das tröstet Sie. In weiser Voraussicht (o wie viele ATARI-Geräte habe ich schon gesehen) wurde der Autor nach einer Stunde Batteriebetrieb sehr vorsichtig...



Bild 2: Die Ports des STACY sind alle hinter Klappen verborgen, die sich aber leicht abnehmen lassen (sollten). Links neben den beiden Buchsen sieht man den Umschalter zwischen Trackball und Maus/Joystick. Ärgerlich ist, daß das Display von seitlich nicht lesbar ist.

Tastenspiele

STACYs Tastatur ist ein klein bißchen kleiner als eine normale ST-Tastatur, soweit es das Haupttastenfeld betrifft. Sie tippt sich aber sehr gut, weil die Tastenkappen oben schmaler sind und dadurch weniger Chancen für Tippfehlerteufel bieten. Die anderen Tasten, Funktions-, Cursor- und Zifferntasten sind viel, viel kleiner. Über den normalen Tasten sind links zwei schmale Reihen von je fünf Funktionstasten, dann, etwas abgesetzt, folgen <Help> und <Undo>. Schließlich: der Cursor-Block auch, ebenfalls chic und zweireihig. Wie ein 'Vatermörder': zu eng, aber es muß sein. Die Tasten oberhalb der 'Buchstaben' sind ja noch ganz brauchbar, sehr eng wird es aber auf dem Ziffernblock weiter rechts. Wohlgemerkt, auf Taschenrechnern sind die Tasten noch kleiner, aber die Zifferntasten haben sehr viel Hub und sind recht schmal. Etwas ungeschickt, vermutlich des einheitlichen Designs wegen. Man kann sich daran gewöhnen. Der Autor, als geübter Gitarrist, trifft schon recht gut (47,5%).

Schließlich gibt es unter dem Ziffernblock eine umgedrehte Maus, einen Trackball. Ab und zu ist dieser etwas hakelig, viel "atariger" (eine Neuschöpfung, die jeder Insider voll Freude begrüßen wird) ist aber, daß die Maus- oder besser Trackball-Knöpfe nur in der Mitte wirklich sicher funktionieren. Am äußeren Rand ist der Doppelklick Glückssache. Aber wie bereits erwähnt, man kann auch eine Maus anschließen. Man kann aber Frieden mit dem Trackball schließen, selbst wenn man (wie der Autor) zu den radikalen Maus-Bevorzugern gehört. Im Kontext (sozusagen) brauchbar. Übrigens kann man am Joystick-Port auch mittels eines Schalters vom Trackball- auf Joystick- bzw. Mausbetrieb wechseln.

Aus reiner Perfidie möchte ich hier z.B. den Mac-Portable nennen: Der umbaubare Trackball für Linkshänder ist eigentlich keine schlechte Idee.

Wenn man als Gulliver bei den Riesen am vorderen Rand der Tastatur stünde, würde einem schwindelig. Die Tastatur ist sehr hoch und entspricht nicht gerade den üblichen Vorstellungen einer ergonomischen Flachtastatur. Des Autors privater Kommentar: Man kann trotzdem sehr gut darauf schreiben. Es ist Platz genug, um die Hände komfortabel aufzulegen, jedenfalls für meine Hände. Auch auf den Knien (nicht Sie, der Laptop auf Ihren) kann man sehr bequem schreiben, und zu schwer ist STACY nicht. Etwas über sechs Kilogramm bringt der Computer auf die Waage, davon geht ein guter Teil auf das Konto der Batterien. Leider werden sie nicht leichter, je leerer sie werden.

Schließlich: Zum Sichtbaren

Richten wir den Blick nach oben zum blau leuchtenden Display. Entgegen anders lautender Gerüchte: STACY hat ein von hinten beleuchtetes LC-Display, dessen Schrift aber leider blau leuchtet, wie bereits anklang. Das Ganze läuft unter der

STACYs Festplatte und Stromverbrauch

Die STACY beherbergt eine SCSI-Platte des renommierten Plattenherstellers Conner. Dessen 3.5"-Laufwerk CP3040 mit integriertem SCSI-Controller verbraucht laut Spezifikation etwa zwei Watt, ist also für Laptops wie geschaffen. Aus der Plattengeometrie mit 1026 Zylindern, 2 Oberflächen und 40 Sektoren pro Spur ergibt sich eine Kapazität von gut 40 MB. Acht kB Pufferspeicher sorgen im SCSI-Controller für zusätzliche Beschleunigung.

Im CHECKHD-Test ergab sich eine Transferrate von 475 kB/s mit bzw. 597 kB/s ohne Zylinderwechsel. Der Plattengeometrie nach zu urteilen, hätte man der Theorie folgend eigentlich 800 bzw. 1200 kB/s erwarten dürfen - ganz offensichtlich läuft die Conner-Platte nur mit Interleave 2 (die gemessenen Werte entsprechen vollständig den theoretisch bei Interleave 2 zu erwartenden), obwohl sie stocksteif von sich behauptet, mit Interleave 1 formatiert worden zu sein. Möglicherweise ist sie das auch, und der Hostadapter bremst; genau war das aber in der kurzen Zeit nicht herauszufinden. Conner behauptet, die Platte liefe auch mit Interleave 1, gibt dazu aber keine Transferraten an.

Die mittlere Zugriffszeit wurde mit 25 bis 27 ms gemessen (offiziell: 25 ms), was ein sehr guter Wert ist. Die Transferrate liegt über typischen Werten für eine SH205 (etwa 410 kB/s bzw. 510 kB/s), aber unter denen einer MEGAFILE 30 (650 kB/s bzw. 780 kB/s). Eine Rakete ist die Platte nicht, aber die guten Zugriffszeiten machen wieder einiges wett.

Die Platte lief leider nicht problemlos mit CBHD, dem Plattentreiber aus SCHEIBENKLEISTER II. CBHD prüft für jede DMA-Adresse, ob ein Untergerät 0 oder ein Untergerät 1 vorhanden ist. Dadurch unterstützt es Zweitlaufwerke in ATARI- und anderen Platten. Wenn man das Conner-Laufwerk der STACY aber nach einem Untergerät 1 fragt, hängt sich dessen Controller auf, alle weiteren Kommandos werden mit Timeout quittiert. Aufgrund dieses Controller-Fehlers ergeben sich, wie man sich vorstellen kann, einige Schwierigkeiten (inzwischen gibt es eine neue CBHD-Version, die dieses Problem umgeht). Beim ATARI-Treiber fällt dieser Fehler nicht auf, weil der nur jeweils ein Untergerät pro Target-Nummer abfragt.

Die Conner-Platte fährt den Drehmotor herunter, wenn man sie parkt; er wird wieder eingeschaltet, wenn sie ausgeparkt wird. Auf diese Weise könnte man Strom sparen - laut Conner verbraucht die Platte in diesem "Standby"-Modus nur etwa 0.5 Watt. Die Hochlaufzeit der CP3040 ist wirklich phänomenal kurz (weniger als fünf Sekunden); STACY und Platte gemeinsam einzuschalten, führt nicht zu Boot-Problemen wie bei anderen STs. Die Platte ist äußerst leise und schon im Bürobetrieb kaum zu vernehmen. Im typischen Laptop-Betrieb unterwegs, ist sie bei typischer Umgebungslautstärke unhörbar.

Gemessener Stromverbrauch am externen Netzteil:

- angenehme Einstellung für Bildhelligkeit und Kontrast, Platte rotiert: 780 mA
- minimaler Stromverbrauch (Bild komplett dunkel, Platte aus): 530 mA
- Stromverbrauch des Diskettenlaufwerks: ca. 150 mA
- Stromverbrauch des Drehmotors der Platte: ca. 120 mA

CB

Rubrik 'naja'. Es ist brauchbar, wenn man Kontrast und Helligkeit voll aufdreht. Leider aber auch nur dann. Besonders bei Batteriebetrieb wirkt sich dies vorteilhaft (aus der Sicht der Batterienhersteller) auf den Stromverbrauch aus. Wie fast jedes LC-Display ist es auch langsamer als ein Monitor; jede Bewegung zieht Schlieren nach sich. Für Spielesessions unterwegs ist STACY also weniger geeignet. Übrigens arbeitet das Display nur im 640*400-Punkte-Modus, die Farbmodi können nicht emuliert werden. Helle Flächen, Fenster oder Menüs zeigen auch leichte helle Schatten. Stört zwar nicht, fällt aber auf. Interessanterweise erzeugt das Display aber Geräusche, ein hohes, leider tönendes (e3 bei voller Intensität) Summen. Mit dem Helligkeitsregler lassen sich hervorragende Düsenjet- und Staubsaugereffekte erzeugen - Fachleute erwarten eine neue Generation von Spielen extra für STACYs Displaysound-Generator.

Rechts neben dem Display sind drei Drehregler für Lautstärke (leider nicht für das Summen des Displays - für den eingebauten Lautsprecher), Helligkeit und Kontrast sowie Kontrolleuchten für Laufwerke und Batterien. Leider ist letztere nicht sehr hilfreich, denn von exakter Kontrolle des Ladezustands kann nicht die Rede sein. Ein bestimmtes Blinkmuster zum Beispiel wäre nicht schlecht oder auch eine Mehrfarben-LED... Der Helligkeitsregler hat eine Doppelfunktion; man kann ihn auch zum Abschalten des Displays verwenden. Ansonsten: Der Tempus-Bildschirmschoner weiß mit dem Display nichts anzufangen, während Protos hervorragend damit fertig wird; es wird schwarz und still.

Über den Drehknöpfen ist sozusagen das 'Gegenüber' (wohl als Arretierung) zum Trackball angebracht. Es ist als eine Art 'Konzepthalter' ausgebildet, als Notizzettel-Halterungsclip. Keine schlechte Idee, auch wenn man keine breiten Formate (zum Beispiel DIN A4-Blätter) daran befestigen kann, ohne die Papiere zu verknittern - der Rand des Display-Gehäuses ist recht hoch.

Gesamturteil zum Display: ok. Für den Preis dürfte es aber besser sein.

Praxis

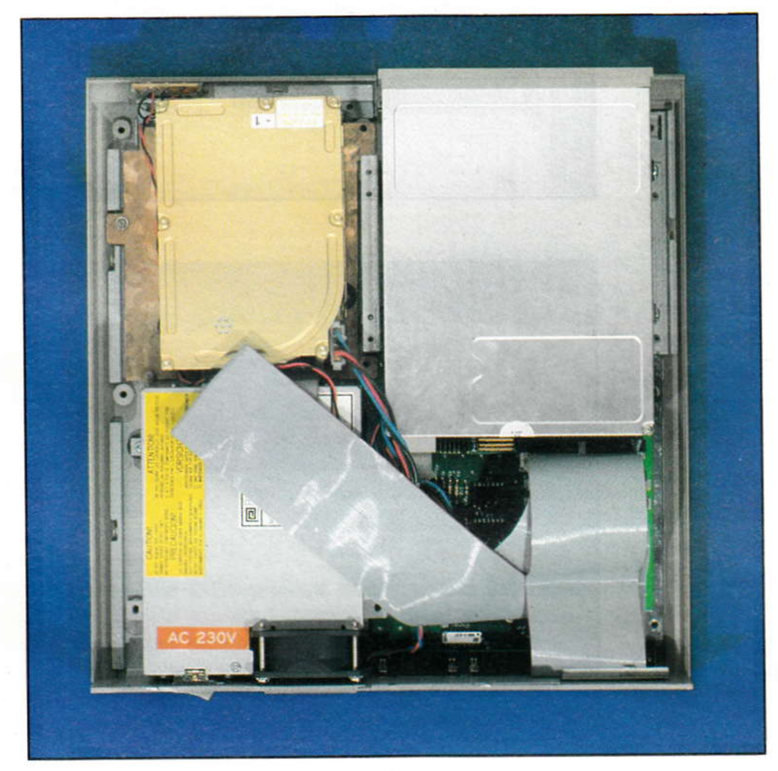
STACY hat nun einiges mitgemacht. Der Autor hat dem Rechner so ziemlich alle schwierigen Programme vorgeworfen, die er finden konnte - außer den üblichen TOS 1.4-Problemen gab es keinerlei Schwierigkeiten. Komplexe Midi-Programme wie Steinbergs Cubase, auch mit Switcher-Programm (um mehrere Programme gleichzeitig im Speicher zu halten) und Timecode-Hardware liefen ebenso zuverlässig wie das Smalltalk 80-System, Calamus, Protos, Signum, TeX usw.

Der Computer begleitete mich auf Reisen (naja, eine Reise, solange durfte ich ihn nicht behalten) zu Literaturrecherchen in die Unibibliothek, in Bussen und Bahnen (auch Autobahnen). Robust sieht er aus, und das ist er wohl auch, jedenfalls verkraftet er durchaus Halbmeter-Stürze. Die Batterien könnten natürlich länger halten, aber es ist wohl völlig gleichgültig, wie lange Batterien überleben, es ist immer zu kurz. In sehr heller Umgebung macht das Display etwas Schwierigkeiten, besonders ungünstig ist Gegenlicht. Je dunkler es ist, desto angenehmer und lesbarer wird die Anzeige, Halbdunkel ist genau das richtige (ein gemütlicher Platz am Kamin zum Beispiel, auf einem Bärenfell). Das einzige, was wirklich stört, sind die ungeschickt konstruierte hintere Klappe und das Display-Geräusch, aber verglichen mit den Geräuschen der 'normalen' Harddisks, mit denen man sich üblicherweise umgibt, ist das nicht weiter schlimm. Der Autor gibt hiermit zu, unverbesserlicher Perfektionist zu sein - schließlich hat so ziemlich jedes Gerät, egal von welchem Hersteller, aus welcher Branche, zu jedem Preis seine konstruktiven Schwächen; nur bei ATARI ist man geneigt, sie ganz besonders sarkastisch zu kommentieren. Das hat wohl seine Gründe.

Auch STACYs Preis ist, verglichen mit PC-Laptops, nicht allzu übertrieben hoch. Ein normaler 286-Computer mit 1 Megabyte RAM und Harddisk kostet im allgemeinen auch über 5000,- DM, und dieser Laptop hat gleich vier Megabyte Speicher. Zur Zeit wird STACY bereits in kleinen Stückzahlen ausgeliefert, große Stückzahlen sind wahrscheinlich erst ab Januar erhältlich. STACY ist ein gelungenes Gerät - chic und praktisch.

CS

Und Sie dreht sich doch



MEGAFILE 44-Tuning

Die Wechselplatte MEGAFILE 44 wurde dem ATARI-User kürzlich als Knüller präsentiert. Die endgültige Lösung stellte sie jedoch nicht dar. Obwohl das Medium "wechselhaft" ist und somit beliebige Speicherkapazitäten erreichbar sind, fehlt es doch an einer guten Backup-Möglichkeit für die ach so wichtigen Daten.

Ein gewisses Trostpflaster stellte das mitgelieferte Backup-Programm von der Firma Application Systems dar; eine schnelle Sicherung, wie sie beispielsweise durch einen Streamer oder eine zweite Festplatte machbar wäre, bringt jedoch ihre Probleme mit sich. Soll man sich für einen weiteren großen Geldbetrag eine zweite Platte anschaffen und sich damit einen fast babylonischen Turm aus SM124, Mega 2, MEGAFILE 44 und MEGAFILE 60 auf das so physikalisch begrenzte Desk, den Schreibtisch stellen? Wer nicht gerade auch noch einen ATARI-Laser sein eigen nennen kann, wird darüber hinaus auch noch Probleme mit dem Sound des Turms haben.

Doch ein Lichtblick schien das amerikanische Originalhandbuch der MEGAFILE 44 zu sein. Hier steht etwas von: "Note: If you have an internal 3 1/2" harddisk installed in your MEGAFILE 44, ..." Doch ein Anruf bei ATARI-Deutschland brachte die Ernüchterung: "In der deutschen Übersetzung wird der Hinweis auf eine interne zweite SCSI-Platte weggelassen und der Einbau auch nicht unterstützt."

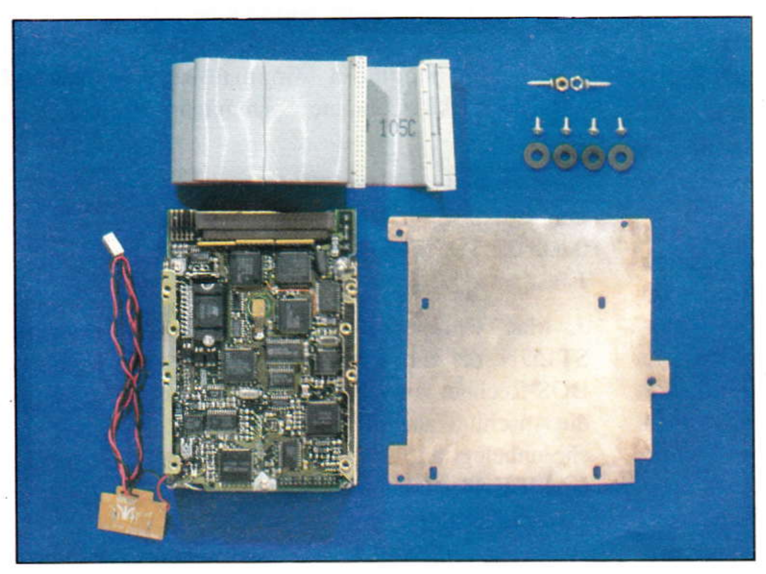


Bild 1: Alle Bauteile (mit Ausnahme des Lüfters), die man zum Umbau benötigt.

Selbst geschickte MEGAFILE 44-Testberichtautoren sagen, es wäre ihnen nicht gelungen eine zweite SCSI-Platte in der MEGAFILE 44 zum Laufen zu bringen. Doch wozu hat ATARI dann alle (teuren) Vorbereitungen getroffen, die zum Einbau einer Zweitplatte notwendig sind? - Die Antwort ist einfach, der Anschluß ist problemlos möglich! Dieser Artikel hier soll nun zeigen, wie mit relativ geringem Aufwand die MEGAFILE 44 zu einer "erwachsenen" Massenspeicherstation aufgebohrt, respektive aufgeschraubt werden kann. Für die bisher so geplagten Ohren ein leiser Papst-Lüfter und für das Mega-Herz eine sehr schnelle 3 1/2" 48 MB SCSI Festplatte. Nun ins Detail.

Der Papst-Lüfter 812L ist überraschend leise und fast vollkommen baugleich mit dem eingebauten Lüfter der MEGAFILE. Vier Schrauben und das Meeresrauschen ist wie verschluckt.

Die Seagate-Festplatte ST157N entkräftet alle Vorurteile gegen Seagate-Platten. Mit einer MTBF von 75000 Stunden und einer mittleren Zugriffszeit von 28 Millisekunden, laut Herstellerangaben, erweisen sich die formatierten 48.6

bzw 46.3 Megabytes (je nachdem, ob 1 Megabyte mit 1000000 oder 1048576 Bytes berechnet wird) als recht brauchbar.

Was muß getan werden, um in den Genuß dieser Vorteile zu kommen? Alle Teile, die in die MEGAFILE 44 eingebaut werden, sind in Bild 1 zu sehen, die ST157N-Festplatte und die Kleinteile. Der Lüfter ist auf dieser Abbildung nicht zu sehen, aber gleich noch zu ihm:

1. Man tausche den Lüfter gegen den Papst-Lüfter aus und löte dessen Kabel an den Stecker, der vorher den anderen Lüfter mit Strom versorgte. Die vier Befestigungsschrauben des alten Lüfters können zur Befestigung des Papst-Lüfters verwendet werden. Falls sich das Netzteilgehäuse als zu knapp erweisen sollte, kann man mit einem mittleren Sandpapier etwas von dem Lüfterrahmen abschleifen. Den Schleifstaub aber unbedingt entfernen!

Bild 3 zeigt das Ergebnis des Umbaus. Die Stomversorgung des alten Lüfters muß noch gekappt werden und das Kabel des neuen Lüfters wird einfach an den Stecker der Lüfterstromversorgung angelötet. Da sich der Stecker leider nicht öffnen läßt, müssen die Kabel des abgeschnittenen Steckers mit den Kabeln des Lüfters verlötet und beide Lötstellen noch mit Isolierband geschützt werden.

2. Man stecke das freie, zweite SCSI-Stromversorgungskabel, welches in der MEGAFILE 44 auf Anschluß wartet, in die Buchse der Seagate-Platte und löse die Verbindung zwischen dem eingebauten SyQest-Laufwerk SQ555 und dem ATARI-Host-Adapter. Auf Stecker J2 kommt nun das Verbindungskabel zur ST157N und auf den Stecker J3 das Verbindungskabel zur SQ555. Beide Kabel sollten lang (30-40 cm) sein. Die Lage der Steckerleisten J2 und J3 auf dem ATARI-Host-Adapter ist in Bild 3 zu sehen. (J2 und J3 sind parallel angeordnet, und J2 liegt nahe an der SQ555.)

Hier noch eine Eigenheit des ATARI-Host-Adapters: Sobald der DIP-Schalter Nummer 1 auf OFF geschaltet wird, wird folgende Logik hergestellt: Die SCSI-Einheit auf Stecker J2 erhält die kleinere Unit-Nummer und die auf J3 die um eins größere. Die Nummer auf J2 wird duch die Einstellung der DIP-Schalter 2 und 3 vorgegeben. Die SCSI-Einheiten müssen ihrerseits auf Unit 0 eingestellt werden, die übrige Adressierung wird durch den ATARI-Host-Adapter vorgenommen.

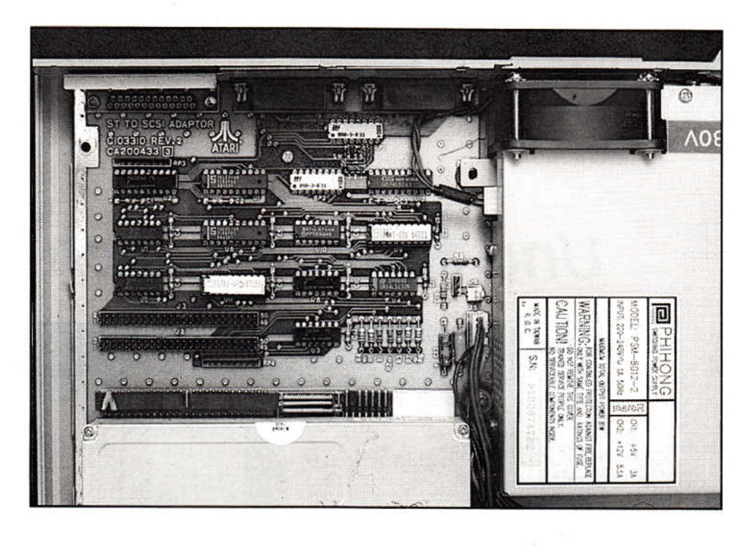


Bild 2: Wo ist das Düsentriebwerk geblieben? - Der leise Papstlüfter und der ATARI SCSI-Adapter.



Bild 3: Der DIP-Schalter auf der Rückseite muß auf die zweite Platte eingestellt werden.

DIP 1	DIP 2	DIP 3	Konfiguration
ON	ON	ON	nur SQ555 ACSI-Unit 0
OFF	ON	ON	ST157N ACSI-Unit 0, SQ555 ACSI-Unit 1
OFF	OFF	ON	ST157N ACSI-Unit 1, SQ555 ACSI-Unit 2
OFF	ON	OFF	ST157N ACSI-Unit 2, SQ555 ACSI-Unit 3
OFF	OFF	OFF	ST157N ACSI-Unit 3, SQ555 ACSI-Unit 4

Tabelle 1: Einstellmöglichkeiten der ACSI-Unit-Nummern

(Die SQ555 steht schon auf SCSI-Unit-Nummer 0, und auch die werksmäßige Adresseneinstellung der ST157N ist 0.) Tabelle 1 zeigt die Möglichkeiten die ACSI-Unit-Nummern einzustellen: (ACSI = ATARI Computer System Interface)

Die in Bild 4 gezeigte DIP-Schalterstellung legt die ST157N-Festplatte auf Unit 0 und die SQ555-Wechselplatte auf Unit 1 fest.

3. Man schraube die Frontplatte der ST157N, die zum Einbau in einen MS-DOS-Rechner vorgesehen ist, ab und löte die Anschlußkabel der gelben LED, welche unbelegt an der Frontseite der MEGAFILE 44 darauf wartet, den Zustand einer eingebauten Festplatte anzuzeigen, auf die Anschlüsse der LED , welche auf der Controller-Platine der SCSI-Platte sitzt. Man sollte noch den Vorwiderstand vor der ATARI-LED überbrücken, da ein solcher schon vor der Platten-LED sitzt. Wichtig ist, hier möglichst kurzzeitig zu löten und am besten einen geerdeten Lötkolben zu verwenden. In der LED der ST157N sieht man bei genauerem Hinsehen einen kürzeren Anschluß und einen längeren. An das Beinchen, das zu dem kürzeren gehört, wird das rote Kabel gelötet, und an das andere Beinchen das schwarze Anschlußkabel. Das Ergebnis zeigt Bild 5, hier ist schon die ATARI-Kontroll-LED an die Platte gelötet, und der helle Stecker, der auf dieser Abbildung noch zu sehen ist, wird wieder auf den Steckplatz J6 auf dem SCSI-Host-Adapter gesteckt (s. Bild 3).

4. Man schneide und bohre sich nach der abgebildeten Vorlage, Bild 6, ein Träger-

blech (1 mm Messing ist mehr als ausreichend) für die SCSI-Platte und montiere die Platte in der MEGAFILE 44. Es ist empfehlenswert, zwischen den Rahmen der SCSI-Platte und das Blech Gummiringe zu legen. (Anstatt Gummiringen kann man auch zerschnittene Gummi-Kabeldurchführungen verwenden.) Man befestige das Blech noch mit zwei Blechschrauben an den von ATARI vorgesehenen Stützen, und der Aufbau ist fertig. Die eingebaute Platte zeigt Bild 6. (Die anderen zwei Befestigungslöcher der Montageplatte werden von den Gehäuseschrauben der MEGAFILE belegt.) Man benötigt

- 4 kleine Blechschrauben, z.B. 2.5 mm*7 mm
- 4 kleine Gummiringe
- 2 kleine Blechschrauben, z.B. 2.5 mm*10 mm
- 2 Unterlegscheiben, z.B. 2 Muttern M4.

Ohne Software ...

... hilft die 'härteste' Hardware nichts.

5. Man gebe die abgemagerte WINCAP-Datei aus Tabelle 2 mit einem Editor ein und kopiere sie zu dem HDX 3.0-Programm auf die MEGAFILE 44-System-Disk.

6. Man wähle nun FORMAT im HDX-Programm und formatiere Unit 0 und Unit 1. (Unit 0 Typ ST157N, Unit 1 Typ MEGAFILE 44) Danach Quit.

7. Mit HINSTALL installiere man nun auf Laufwerk C (ST157N) den Plattentreiber. Diese Standardpartitionierung soll folgende logische Laufwerke erzeugen:

Festplatte: ST157N (46.3 MB)

C: 4 MB Systemdateien und nicht veränderliche Daten (z.B.Programme)
D: 14 MB veränderliche Daten/Programme

E: 14 MB veränderliche Daten/ Programme

F: 14 MB veränderliche Daten/ Programme

Wechselplatte: SQ555 (42.3 MB)

G: 14 MB Backup von D:
H: 14 MB Backup von E:
I: 14 MB Backup von F:

Die ST157N wurde absichtlich auf Unit 0 gesetzt, damit sie als Boot-Platte ihre Partition C: zur Verfügung stellen kann. Andernfalls müßte man auf jedes Wech-

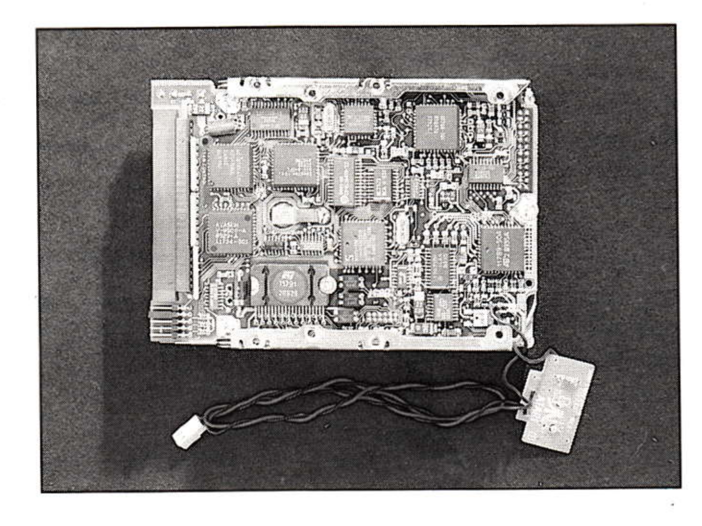


Bild 4: Die Anschlußkabel der in die MEGAFILE 44 eingebauten Kontroll-LED muß auf die Anschlüsse der kleinen LED auf der ST157N gelötet werden.

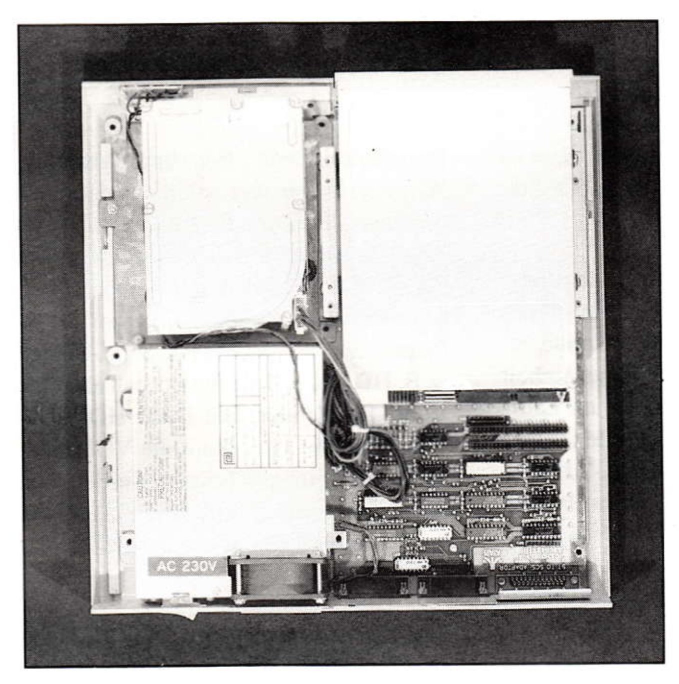


Bild 5: Die Platte ist eingebaut, es fehlen nur noch die Kabel.

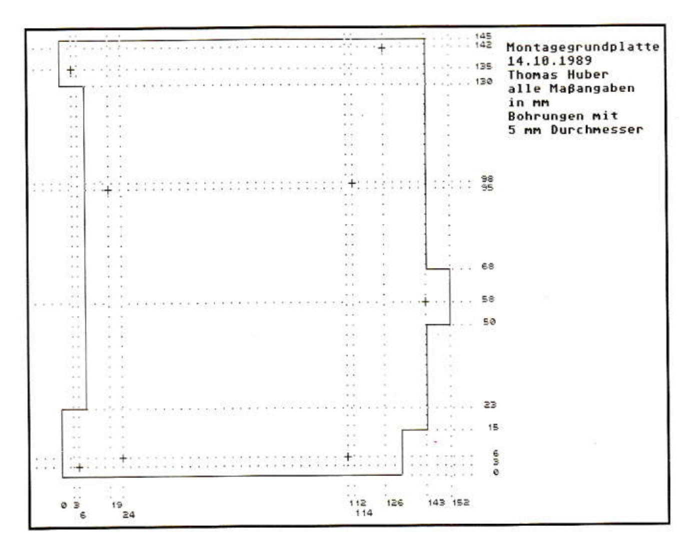


Bild 6: Montageplatte - nach diesem Plan muß die Kupferplatte zugeschnitten und gebohrt werden.

```
# Hard disk format and partition configuration file.
# (WINCAP Datei)
# 11-Okt-1989       TH-Software
:@@=Parameter,s:ms#32760:
44 Mb :mn=MEGAFILE 44:md#0:pt=14-14-14:dp#0x6333:
46 Mb :mn=ST-157N          :md#0:pt=4-14-14-
14:dp#0x6333:
:46=4-14-14-14:p0#4m:p1#14m:p2#14m:p3#14m:
:42=14-14-14:p0#14m:p1#14m:p2#14m
```

Tabelle 2: Die abgemagerte WINCAP-Datei

selmedium für die SQ555 eine Boot-Partition einrichten, was bei einer komfortablen Arbeitsumgebung einiges an Speicherplatz verbraucht. Falls man jedoch verschiedene Anwendungen hat und für jede eigene Accessories, Desktop usw. braucht, ist es empfehlenswert, die SQ555 als Boot-Platte Unit 0 zu verwenden und die ST157N nur für immer gebrauchte Daten und Programme abzustellen.

Natürlich kann die Wechselplatte auch als Festplatte anstatt als Backup-Medium gebraucht werden. Eine Sicherung der Daten ist unter diesen Umständen allerdings recht schwer, da zuerst auf Festplatte und dann wieder auf Wechselplatte gesichert werden muß, aber es gibt ja auch noch die Methode, auf Disketten zu sichern.....

Die Seagate-Platte hat noch den von VORTEX-Platten bekannten Vorteil des Auto-Parks. Die SyQest-Platte muß jedoch vor Abschalten der Platteneinheit durch Drücken der Stop-Taste am Laufwerk heruntergefahren werden. Das SHIP-Programm auf der HDX 3.0-Diskette erfüllt jedoch alle notwendigen Funktionen: Die SQ555 wird gestoppt und die ST157N geparkt. Das SHIPpen des Massenspeichers hat den Vorteil, daß nach Neueinschalten des Systems nicht mehr der Auswurfhebel der Wechselplatte "eingekuppelt" werden muß, sondern alles geschieht automatisch.

```
HDPLUS.TYP

Externe Steuerdaten für das Utility HDPLUS.PRG
(HDPLUS.TYP Datei)
Anpassung an das SQ555 und ST157N Laufwerk
11-Okt-1989 TH-Software

typ(0)=hd40       name(0)=' ST157N'
typ(1)=hd40       name(1)=' SQ555 '

cyl(0)=0  hds(0)=0  rwc(0)=0  pre(0)=0  spt(0)=0
sek(0)=0
cyl(0)=0  hds(0)=0  rwc(0)=0  pre(0)=0  spt(0)=0
sek(0)=0

cpy(0)=94850
cpy(1)=86690

def_part(0,0)='4-14-14-
14',8834,28672,28672,28672;
def_part(1,0)='14-14-14',28896,28896,28896;

soft_park(0)
soft_park(1)
```

Tabelle 3: Die Anpassung der HDPLUS.TYP-Datei

Die ATARI-Platte geht fremd

Die Zeiten, als die Programme mit ATARI- oder VORTEX-Platten bzw. deren Treibern keine Schwierigkeiten hatten, sind nun leider vorbei, das HDX 3.0 bricht den de facto-Standard, und so wichtige Programme wie Backupprogramme, Netzwerktreiber usw. streiken angesichts des neuen Treibers. Es gibt hier eine elegante Lösung für Leute, die im Besitz einer VORTEX-Platten-Software sind. Man definiere die beiden neuen SCSI-Platten als alte VORTEX-SCSI-Platten (z.B. HD40), und sie fühlen sich als reine VORTEX-Kinder. Die Datei HDPLUS. TYP (Tabelle 3) muß auf die VORTEX-Systemdiskette geschrieben werden, und es kann losgehen; Formatieren, Autoboot und Parken geschieht wie bei VORTEX-Platten.

Und der Bonus: Auf den VORTEX-Systemdisketten gibt es ein rasantes Partition-Backup-Programm (BACKPART.PRG), welches auch mit den 'Pseudo'-VORTEX-Platten einwandfrei funktioniert. Dieses Backup-Programm läßt sich mit folgender Batch-Datei versehen und sichert automatisch nach Aufruf von BACKPART.PRG die drei (s.oben) Datenpartitionen der ST157N auf die Wechselplatte. Viele andere Backup-Programme verfügen nicht über die Möglichkeit, eine Batch-Datei einzugeben.

```
BACKPART.BAT für BACKPART.PRG

D G N
E H N
F I N
.
```

(Erläuterung: Sicherung von Partition D auf G, E auf H und F auf I. Sicherheitsabfragen werden durch das N verhindert. Der Punkt zeigt das Dateiende an.)

Das programmgesteuerte mechanische Abschalten des Wechselplattenlaufwerks läßt sich mit dem SHIP.PRG des HDX 3.0 erreichen, auch wenn die Platten VORTEX-programmiert sind.

Doch nichts ist ohne Fehler!

Viele Programme haben mit dem HDX 3.0 Kompatibilitätsprobleme. Der Nachteil des VORTEX-Plattentreibers ist, daß der Medienwechsel nur nach RESET des Computers anerkannt wird und auch keine MS-DOS-beschriebenen Wechselplatten gelesen werden können. Doch wer hat noch einen PC mit SyQuest-Wechselplatte? Wer jedoch nur eine schnelle Festplatte mit einer schnellen Sicherungsmöglichkeit sucht, kann gut auf den VORTEX-Treiber zurückgreifen. Und nun viel Spaß beim "Aufbohren". Falls zu dem Einbau noch Fragen auftauchen, richten Sie sich bitte schriftlich an:

Thomas Huber
Von der Pfordtenstraße 29
8000 München 21

Literatur:

[1] Brod: Wechselhaft; Die MEGAFILE 44 im Test, ST-Computer 9/89, S.25ff

[2] Jankowski, Reschke, Rabich: ATARI ST Profibuch, 2.Auflage, Sybex Verlag 1989

Der PC-Emulator für alle ATARI-Computer*

SuperCharger**



* Mit Prozessoren der 68000er Familie und TOS

DM 798,–

Einfach
Einfachste Installation durch Anschluß an den DMA-Port des ATARI. Keine Werkzeuge etc. notwendig. Kein Garantieverlust durch Öffnung des ATARI zu befürchten.

Sicher
Durch den externen Anschluß und die durch Fertigung bei SIEMENS erreichte hohe Qualitätskontrolle des SuperCharger wird eine Beschädigung Ihres ATARI ausgeschlossen.

Flexibel
Der SuperCharger kann ohne weiteres transportiert und so an mehreren ATARI verwendet werden. Auch bei Kauf eines weiteren ATARI der neuesten Typen kann der SuperCharger weiter verwendet werden.

Schnell
Durch eine optimierte Programmierung des eigenen BIOS wird eine extrem hohe Geschwindigkeit erreicht, die keineswegs das Gefühl aufkommen läßt, an einem Emulator zu arbeiten. Das emulierte Original wird in Bezug auf Geschwindigkeit weit übertroffen.

Komfortabel
HOTKEY ermöglicht das „Einfrieren" des laufenden Programms, Arbeiten unter TOS und anschließende Rückkehr zu MS-DOS. Der ATARI Laserdrukker kann unter DOS verwendet werden. Auch unter DOS sind zweiseitige ATARI-Disks lesbar. Der 8087 Coprozessor wird voll unterstützt.

Komplett
Wir machen Nägel mit Köpfen – im Lieferumfang enthalten sind: MS-DOS 4.01, 512 KB RAM, deutsche Installations- und Bedienungsanleitung.

** Ein Produkt der Beta Systems Computer AG, Ffm.





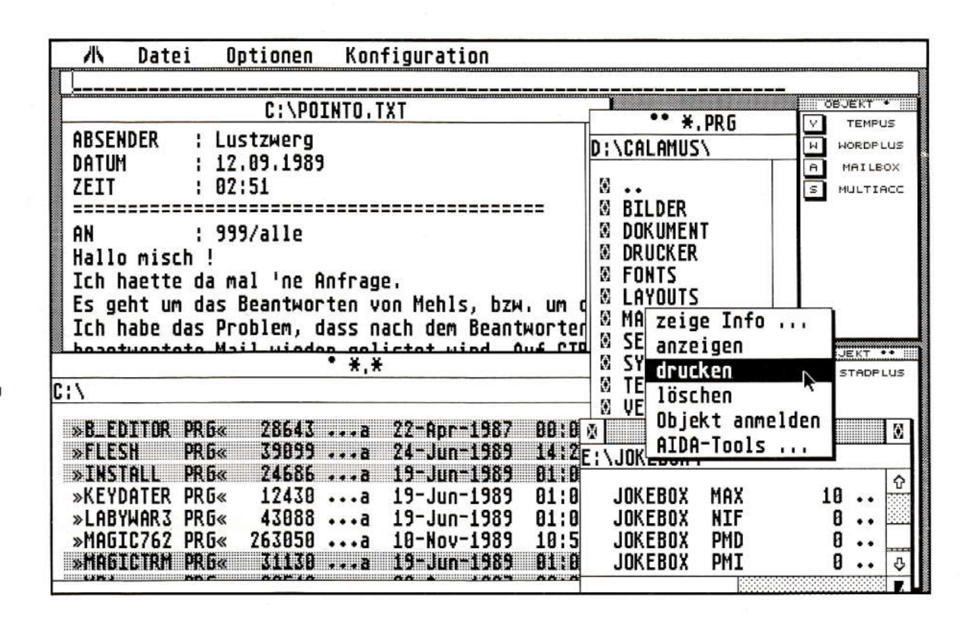
Exklusivvertrieb Zentraleuropa:
SEH Computer-Peripherie-Geräte GmbH
Beethovenstraße 26
6455 Erlensee
Tel: (061 83) 83-0, Fax: (061 83) 83 38

Vertrieb Handel:
Hako AG
Burgstraße 23-25
4630 Bochum 6
Tel: (023 27) 303-0, Fax: (023 27) 303 134

Die SEH ist außerdem vertreten in Düsseldorf · Nürnberg · Augsburg · Stuttgart · München

AIDA

Die neue Shell-Dimension



Die Firma "MK SOFT EDV" wirbt mit "Gib AIDA eine Chance" für Ihre Shell für den ST. Mag man sich auch über den Werbespruch aufregen; wir haben AIDA eine Chance gegeben - das Programm hat sie auch gut genutzt.

AIDA ist sowohl eine grafische Shell als auch ein Command-Line-Interpreter für den ST. Wen hat die karge Benutzerführung mit den überaus informativen Monologen des ST ("Diese Anwendung kann das angegebene Objekt nicht finden", "Drucker streikt ?!?") schließlich noch nicht gestört? AIDA ("Advanced Integrated Desktop Application") versucht nun geschickt, dieses Manko abzubauen.

Der Start

Nachdem AIDA gestartet ist, präsentiert es sich wie in Bild 1. Die Oberfläche scheint zwar auf den ersten Blick bekannt zu wirken, hat aber einige entscheidende Verbesserungen erhalten. Zum einen können endlich auch Programme auf das Desktop gelegt werden, um sie von dort aus starten zu können. Doch damit nicht genug: Man kann jedem Programm auch ein Tastaturkürzel zuweisen, mit dem es gestartet werden kann - Tempus mit ALT-T zu starten, ist schon eine ungeheure Erleichterung.

Doch nicht nur Programme, sondern auch (fast) alle gewohnten Desktop-Operationen können mit Tastaturkürzeln, diesmal jedoch mit Control, durchgeführt werden. Sogar Disketten können sich durch einen Tastaturdruck formatieren oder kopieren lassen. Die Wahl der Codes ist dabei allerdings etwas unglücklich geraten, da die Kürzel nicht sehr leicht zu merken sind (Formatieren = CTRL-A, Objekt anmelden = CTRL-L), doch mit ein wenig Übung hat man die Hürde gemeistert.

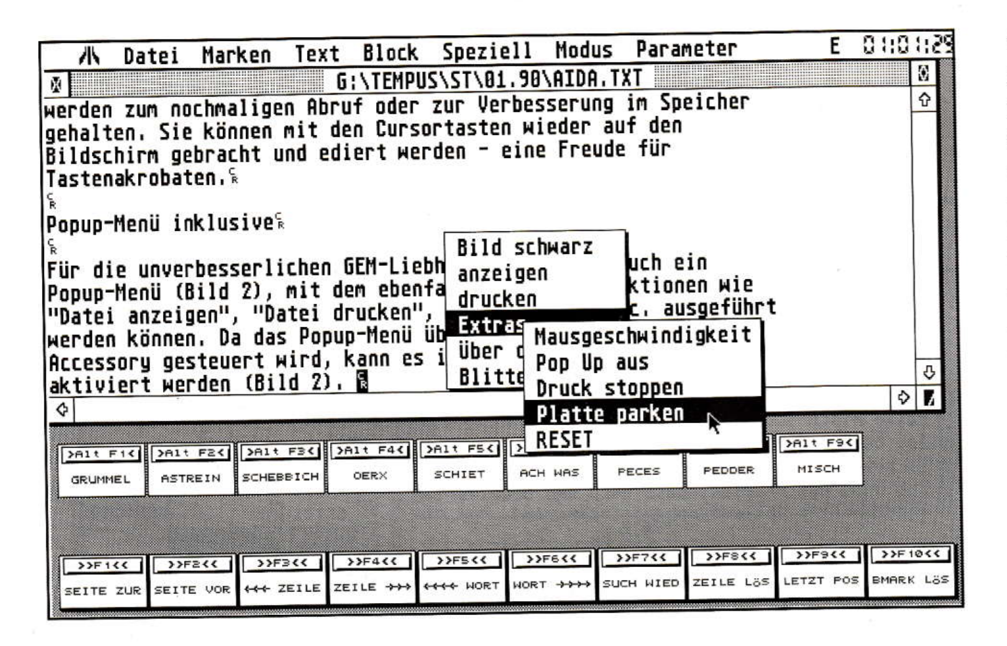
Ja, wo laufen sie denn?

Beim Blick auf das neue Desktop (Bild 1) fällt auf, daß nirgendwo ein Laufwerkssymbol zu finden ist. Wie läßt man nun aber ein Window mit einem Inhaltsverzeichnis anzeigen? Kein Problem, dazu stehen dem Benutzer drei Möglichkeiten zur Verfügung. Entweder durch einen Druck auf CTRL-O, worauf man dann das Laufwerk auswählen kann. Mag man lieber die Maus, genügt auch ein Doppelklick auf das graue Desktop. Auch hier ist dann das Laufwerk auszuwählen. Wenn Sie lieber Kommandos eingeben, können Sie äquivalent auch "dir c:*.*" eintippen, um somit beispielsweise Laufwerk C: anzeigen zu lassen. Anhand des Kommandos "dir" läßt sich schon erahnen, daß AIDA bei seinen Befehlen an MS-DOS angelehnt ist. So stehen ebenfalls die Befehle cd, copy, dir, erase, label, move, quit, restart, stop, chdir, date, diskcopy, format, md, path, ram, size, type, cls, del, echo, goto, mkdir, print, rename, status und wait zur Verfügung, mit denen sich schon recht vernünftig arbeiten läßt. Bis zu 10 Befehle, die in der Kommandozeile eingegeben wurden, werden zum nochmaligen Abruf oder zur Verbesserung im Speicher gehalten. Sie können mit den Cursor-Tasten wieder auf den Bildschirm gebracht und ediert werden - eine Freude für Tastenakrobaten.

Popup-Menü inklusive

Für die unverbesserlichen GEM-Liebhaber existiert auch ein Popup-Menü (Bild 2), mit dem ebenfalls wichtige Funktionen wie "Datei anzeigen", "Datei drukken", "Info zeigen" etc. ausgeführt werden können. Da das Popup-Menü über ein (mitgeliefertes) Accessory gesteuert wird, kann es in (fast) jedem Programm aktiviert werden (Bild 2).

Einige GEM-Operationen haben sich bei AIDA etwas verändert, so daß man viele gewohnte Arbeitsgänge umstellen muß. So wird bei AIDA durch Druck auf das Schließsymbol eines Fensters nicht eine Directory-Ebene höher gewechselt, sondern sofort das gesamte Fenster geschlossen, was ja eigentlich auch logisch ist bei einem Schließsymbol. Eine Directory-Ebene höher gelangt man, indem man auf das Verzeichnis ".." klickt oder CTRL-Up drückt. War man es gewohnt, beim Druck auf das "Full"-Symbol eines Fensters sofort mit einem Riesenfenster übermannt zu werden, freut man sich bei AIDA angenehm: Das Fenster wird hier auf eine sinnvolle Größe gebracht. Der waagerechte Slider, der im normalen Desktop bislang schmerzlich fehlte, ist nun auch endlich vorhanden.



Ein anderes Manko, das bislang in fast jedem ATARI-GEM-Programm nervte, war, daß man zwar eine Selektion in einem Fenster machen konnte, diese jedoch rückgängig gemacht wurde, sobald beispielsweise das Fenster verschoben wurde. Diesem Übel bereitet AIDA den Garaus und bietet dabei sogar noch mehr. Sind schon Dateien selektiert, kann auch dann noch mit dem Gummiband gearbeitet werden: neu selektierte Dateien werden hinzugefügt, doppelt selektierte Dateien werden deselektiert. Danach kann das Fenster natürlich auch verschoben werden, die Selektion bleibt bestehen. Lediglich beim Öffnen eines neuen Fensters werden alle selektierten Dateien wieder deselektiert. Vorteilhaft: durch Doppelklick in den leeren Bereich eines Fensters wird alles selektiert.

Batch-Dateien

Die von MS-DOS und anderen Betriebssystemen her bekannten Batch-Dateien, die einen vollständigen Arbeitsprozeß steuern können, sind auch bei AIDA möglich. Dabei können alle Befehle genutzt werden, die auch im Command-Line-Editor eingegeben werden können. Außerdem kann in den Batch-Dateien selbst mit Labels gearbeitet werden, die bedingt oder unbedingt angesprungen werden können. Natürlich können einer Batch-Datei auch Parameter übergeben werden. So ist es dann beispielsweise möglich, eine Textverarbeitung aufzurufen, der gleich der Name der zu bearbeitenden Datei übergeben wird. Man könnte zum Beispiel auch beim Verlassen der Textverarbeitung gleich alle "*.DUP"- bzw. "*.BAK"-Dateien löschen, damit die Festplatte immer übersichtlich bleibt. Bis zu fünf Batch-Dateien können ineinander verschachtelt werden.

Mehr!

Wenn Ihnen auch diese Funktionen noch nicht genügen, soll noch eine genannt sein. Ist es Ihnen auf dem Desktop noch nicht passiert, daß Sie sich eine Datei mit 0 Bytes Länge anzeigen lassen wollten? Was passiert, ist bekannt: ST grüßt Manitou. Mit AIDA kann Ihnen das nicht mehr passieren, da alle Dateien, die Sie sich anzeigen lassen, in einem Textfenster angezeigt werden. Der Vorteil liegt auf der Hand: Sie können weiterarbeiten, während das Textfenster geöffnet ist. Dadurch ist es möglich, ständig auf den Text zurückzugreifen. Ein kleiner Nachteil sollte allerdings erwähnt werden: das Scrolling im Textfenster ist wesentlich zu langsam. Ein weiterer Nachteil: Wenn Sie eine 280k-Textdatei einladen und den horizontalen Slider dann an das untere Ende des Rollbalkens schieben, dauert es knapp eine Minute, bis der untere Text erscheint. Das ist das einzige grundlegende Manko, das ich Ihnen nennen kann.

Für wen?

Natürlich ließen sich auch alle Möglichkeiten kombinieren. So ist es möglich, eine Batch-Prozedur per Tastendruck aufzurufen. Aus diesen Möglichkeiten, die längst noch nicht alle sind, ist bereits zu ersehen, daß mit AIDA sehr gut zu arbeiten ist. Wenn Sie allerdings kein Freund von vielen Tastendrücken sind und lieber mit der guten, alten Maus arbeiten, sollten Sie sich lieber nach einer anderen Shell umsehen. Die Benutzung von AIDA ist tatsächlich nur sinnvoll, wenn man lieber mit der Tastatur arbeitet, obwohl es auch hier manchmal nerven kann, hin und wieder zur Maus greifen zu müssen. AIDA kostet DM 147,-. Der Preis ist für das Programm angemessen, für sein Geld bekommt man auch Leistung geboten.

MP

Bezugsquelle:

MK SOFT EDV Beratung GmbH
Sudetenstraße 39
6842 Bürstadt
Tel. (06245) 7070

SHERLOOK

Dem Text auf der Spur

Auch wenn der Vergleich etwas hinkt, mit detektivischer Kleinarbeit hat Schrifterkennung durchaus etwas zu tun. (Man erinnere sich an Artikel in früheren Ausgaben dieser Zeitschrift.) Dann denke man z.B. an Hieroglyphen: Aus vielen Spielfilmen ist das Bild geläufig, wenn Forscher in Ägypten mittels eines Vergrößerungsglases Inschriften in Grabmauern zu ergründen versuchen. Im Grunde tut unser Computer bei der Schrifterkennung gar nichts anderes.

Also paßt das Bild durchaus, wenn man sich vorstellt, daß Sherlock Holmes als Software (Sir Arthur Conan Doyle möge verzeihen) in den RAMs unseres ST tätig wird. Ein ähnliches Bild hatten wohl die Schöpfer von "*SHERLOOK*" vor Augen, als ihre Schrifterkennung fertig war.

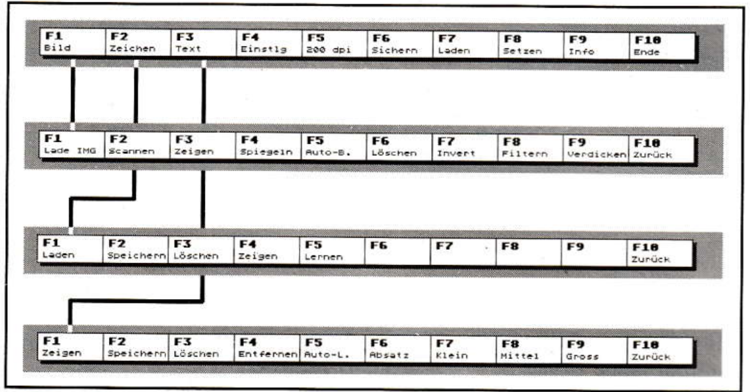


Bild 1: SHERLOOK benutzt keine Pull-Down-Menüs! Dennoch sind alle Aktionen per Maus (als Buttons) oder "von Hand" als Funktionstasten anwählbar.

SHERLOOK liegt uns in der Version 2.3 vor und besteht aus einer Diskette, einem schmalen Handbuch und einem "*Dongle*". Letzteres bedarf einer kleinen Erklärung: Aus weithin bekannten Gründen gehen die Entwickler hochwertiger Software mehr und mehr dazu über, nun doch wieder einen Kopierschutz in die Programme einzubauen. Eben dieser "*Dongle*", entweder als ROM-Port- oder als Joystick-Port-Stecker ausgelegt, stellt den Kopierschutz dar. Man kann lange über Sinn und Unsinn solcher Schutzmaßnahmen streiten (wird, glaube ich, auch heute noch heftigst getan). Gerade aber, wenn es darum geht, neue Anwendungsgebiete mit einem Programm zu erschließen, gewissermaßen Vorkämpfer auf unbekanntem Terrain zu sein, scheint mir persönlich der Einsatz solcher Kopierschutzstecker für eine gewisse Zeit und ab einem bestimmten Preisniveau gerechtfertigt zu sein. Trotz dieser Ausführungen möchte ich die Entwickler dazu anregen, sich Gedanken über andere Schutzmethoden zu machen. Ein Gutes aber hat die "Dongle-Zugabe" bei *SHERLOOK*, wer schon einen solchen Schutzstecker anderer Firmen besitzt (sogenannte "*ST-LOCK-Module*"), kann *SHERLOOK* auf diese anpassen lassen. Damit laufen alle Programme mit nur einem Stecker. Außerdem bemüht sich die Firma H. Richter, Gevelsberg, um eine Standardisierung der ST-LOCK-Technik, so daß bei nachfolgenden (geschützten) Programmen keine weiteren Stecker mehr nötig werden.

Wie war das noch gleich? Man schafft sich einen Scanner an, um nicht mehr länger Texte abtippen zu müssen? Oder hat man schon einen Scanner für Grafik, dann wäre doch die automatische Texterfassung ein interessantes neues Tätigkeitsfeld für das Lesegrät? Sehen wir uns doch *SHERLOOK* näher an:

Nach dem Programmstart läuft ein kleiner Film ab. Die Titelzeile rollt von rechts herein, und ein Begrüßungsbild "pixelt" sich in der Mitte auf. Weil das nicht jedermanns Geschmack ist, kann diese Einstiegszeremonie per >*ESC*< sinnvollerweise auch umgangen werden.

Wenn dann die Arbeitsoberfläche erscheint, fällt eines dem GEM-geübten Auge sofort auf: Die Pull-Down-Menüs fehlen! Aber was soll's, mit der Maus kann man trotzdem weiterarbeiten, denn die Funktionstastenleiste am unteren Bildrand ist auch per Maus zugänglich. Und seien wir mal ehrlich, ohne Pull-Downs geht es doch allemal schneller.

Im Hauptfenster finden wir rechts die verkleinerte Originalvorlage des eingelesenen Bildes und links einen (um den Faktor acht) vergrößerten Teilausschnitt. Das Komplettbild rechts paßt sich maßstabsgerecht dem Originalformat an. Eine Informationsleiste links zeigt wichtige Systemzustände an.

Bitte ein Bild!

Üblicherweise wird ein Programm zur Schrifterkennung gleichzeitig mit einem Scanner betrieben. Ganz unüblich muß-

ten bis vor kurzem Besitzer eines Handyscanners arbeiten (so auch ich), denn im ROM-Port wollte sowohl der Handy- als auch der SHERLOOK-Dongle stecken. Da aber mehrheitlich der "übliche Weg" gegangen wird, steuert *SHERLOOK* folgende Scannertypen direkt an:

- Hawk CP14
- Print Technik Universal
- ITD-Scanner
- SPAT
- Panasonic 505/506 mit MARVIN-DMA-Interface
- Epson Farbscanner

(Weitere Treiber kommen alsbald hinzu.)

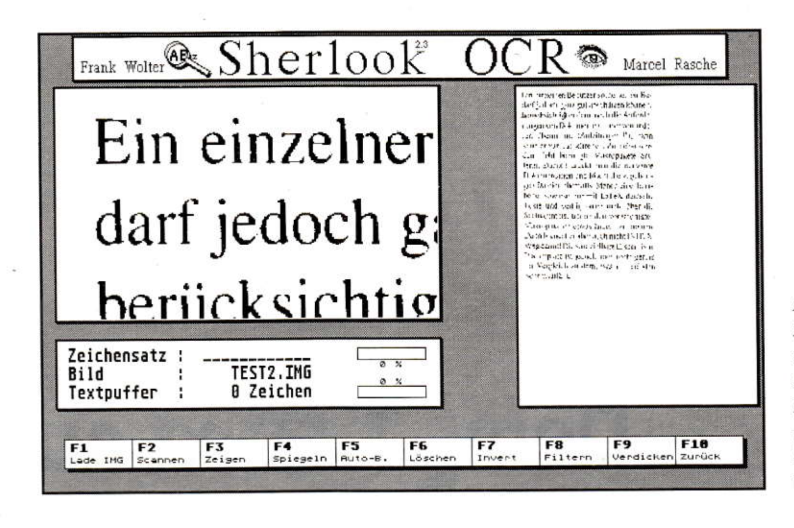
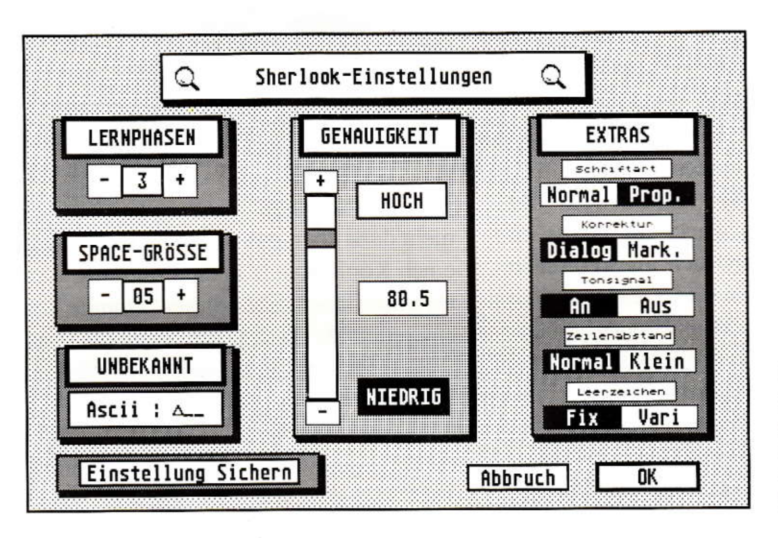


Bild 2: Das Hauptarbeitsfenster zeigt rechts die Vorlage in Komplettgröße und gleichzeitig links "lesbar" vergrößert.

Durch Druck auf die Tasten F1 (Bild) und F2 (Scannen) erscheint ein Kontrollfenster für Scanner-Einstellungen. Jetzt können Parameter für den nächsten Scandurchgang festgelegt werden wie Graustufen, Auflösung oder Koordinaten für Teilausschnitte (z.B. bei Panasonic). Der eigentliche Scan-Vorgang wird nun mit >*Return*< gestartet. Sobald das komplette Bild eingelesen ist, baut es sich im rechten Teil der Arbeitsoberfläche auf.



Vielfältige Einstellungen erlauben Manipulationen des eingescannten Bildes.

Mit den Tasten F1 (Bild) und F1 (Lade IMG) kann alternativ auf bereits abgespeicherte Bilder anderer Scan-Programme (der unübliche Weg, s.o.) zugegriffen werden.

Dies war die erste Hürde. Bevor nun mit dem Erkennvorgang begonnen wird, sollte die Qualität der Vorlage geprüft werden. Mit dem rechten Komplettbild erkennt man großflächige Fehler wie Kontrastunterschiede, Blendstreifen durch Lichteinfall, sehr schiefe Zeilenführung. Mit dem linken Vergrößerungsausschnitt werden Unzulänglichkeiten der Buchstaben deutlich. Am besten den Scan-Vorgang dann mit veränderten Parametern wiederholen.



SHERLOOK im Erkennungsmodus. Der Zeichensatz links unter dem Textausschnitt ist per Tastatur "erlernbar", ein erweiterter Zeichensatz im untersten Rahmen steuert man per Maus an.

Manipulationen beim Lesevorgang

Nicht in allen Fällen muß der Scan-Vorgang mehrmals wiederholt werden, bis ein befriedigendes Bild vorliegt. *SHERLOOK* stellt diverse Funktionen zur Verfügung, um Veränderungen im Bild vornehmen zu können.

Vorlage verkehrt eingelesen? Kein Problem - *SHERLOOK* erlaubt die Spiegelung der Vorlage horizontal, vertikal oder beides gleichzeitig.

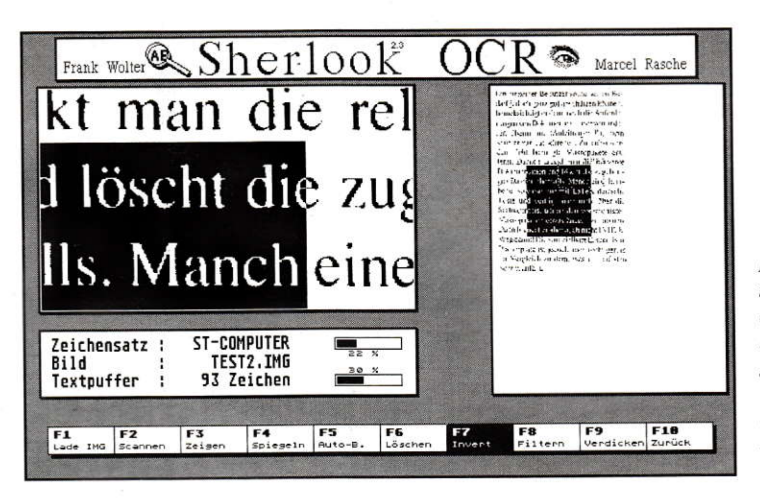
Teilausschnitt ignorieren? Wenn bestimmte Textteile für den Erkennungsprozeß ohne Bedeutung sind, können diese durch Invertieren (erscheint dann als schwarzes Rechteck) gesperrt werden.

Teile völlig herausnehmen? Mit der Funktion *"Ausschnitt löschen"* verschwinden definierte Teile völlig aus der Vorlage und fallen dem Erkennungsvorgang später nicht zur Last.

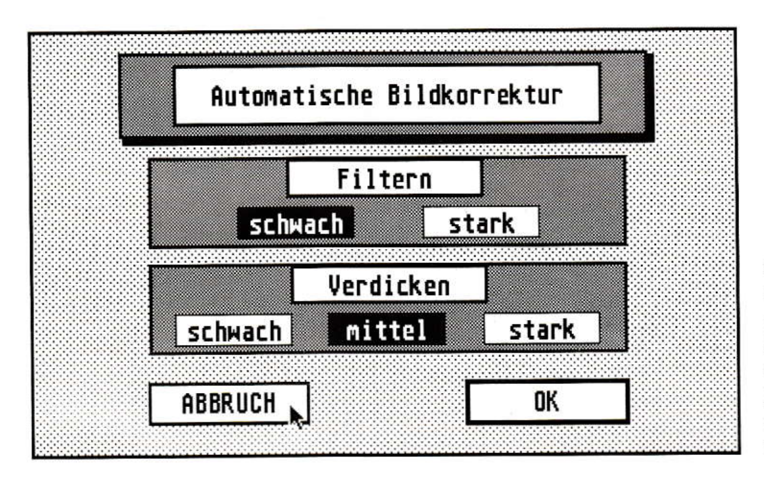
Vorlage filtern? Bei einer schlechten Vorlage können sehr schnell Ausfransungen der Buchstaben entstehen. Dies insbesondere dadurch, weil immer ein Kompromiß zwischen Helligkeit und Kontrast eingegangen werden muß. Dann treten auch "Fehlfarben" wie Graustiche oder Kontraststreifen auf. Mit zwei Filterfunktionen kann dies weitgehend bereinigt werden (im wahrsten Sinne des Wortes): *Filter 1* (schwach) zur Tilgung einzelner

angehängter Bildpünktchen oder *Filter 2* (stark), um Linien und Striche zu eliminieren.

Ausschnitt verdicken? *SHERLOOK* funktioniert immer besser, je dicker und kräftiger die Buchstaben sind. Aber Achtung! Nicht allein deswegen Helligkeit und Kontrast des Scanners bis "*maximal*" aufdrehen. Durch Verdicken werden die Buchstaben im nachhinein künstlich aufgebläht. Dies kann durchaus notwendig sein, wenn die Zeichen zu dünn oder zweigeteilt sind.



Mit einem invertierten Rahmen (schwarz unterlegt) können Textpassagen gesperrt werden. Diesen Teil ignoriert SHERLOOK im Erkennungsmodus.



Selbst wenn das Bild schon eingescannt ist, können nachträgliche "Schönheitsoperationen" durchgeführt werden.

Kennen Sie den?

Die zentrale Funktion in *SHERLOOK* ist das Rahmenziehen mit der rechten Maustaste. Sobald ein Rahmen aufgezogen ist, wird darin der Erkennungsprozeß eingeleitet. Maximal sind 32 solcher Rahmen in einer Vorlage möglich.

Mit den Tasten F2 (Zeichen) und F5 (Lernen) kommt man nun in die Verwaltung des Zeichensatzes und damit zum Lernen der unbekannten Zeichen. Hierzu erscheint eine etwas anders gestaltete Arbeitsoberfläche. Der Text im linken Ausschnitt behält Lage und Form wie vorher. Nur im rechten Teil wird jetzt das erste unbekannte Zeichen noch einmal (etwa um den Faktor vier) vergrößert dargestellt.

Jetzt soll das Zeichen per Tastendruck quasi bestätigt werden. Alle Zeichen, die per Tastatur zugänglich sind (sogenannter *Standardzeichensatz*), findet man links unter dem Textausschnitt abgebildet. Sollten darüber hinaus Sonderzeichen oder nationale Schriften verlangt sein, wären diese per Maus in der Leiste am unteren Bildrand anzuwählen.

Manipulationen beim Erkennungsvorgang

Bevor überhaupt ein Erkennungsdurchgang startet, erscheint eine Kontrollbox für Parametereinstellungen. Dort wird festgelegt, wie Schriftart (*proportional* oder *Schreibmaschine*), Zeilenabstand (*normal/klein*) oder Wortzwischenräume (immer nur *eine Leerstelle* oder *variabel* wie bei Blocksatz) zu erwarten sind. Besonders interessant sind aber Genauigkeitsregler und die Einstellung der Lernphasen.

Mit dem Toleranzregler für die Fehlergenauigkeit wird festgelegt, wie groß die Abweichungen sein dürfen, damit ein Zeichen noch zu ähnlichen zugeordnet werden darf. Ist die Toleranz hoch (also sehr empfindlich) eingestellt, wird das Programm öfters nachfragen, um Ähnlichkeiten vom Benutzer per Tastatur bestätigt zu bekommen. Je niedriger diese Toleranzschwelle angesetzt ist, desto eher akzeptiert der Rechner gröbere Ähnlichkeit. Die Wahl des Toleranzwertes sollte auch abhängig gemacht sein von der Qualität der Vorlage - je besser die Vorlage, desto niedriger die Fehlergenauigkeit.

Sehr interessant sind die Lernphasen, einstellbar von eins bis zehn. Wenn bei der Funktion "*Zeichen lernen*" vom Benutzer per Tastatur die Buchstaben eingetippt werden, merkt sich das System diese und markiert sie in der Leiste des Standardzeichensatzes grau. Aber dieser eine Buchstabe genügt noch nicht, um späterhinaus dem System hinreichende Sicherheit für ähnlich aussehende Zeichen zu geben. Deswegen verlangt *SHERLOOK* öfters ähnliche Zeichen per Tastatur zu bestätigen - selbst wenn der Benutzer kaum Unterschiede ausmachen konnte. Die Zahl der Lernphasen legt nun fest, wie oft dasselbe Zeichen bestätigt werden muß, um endgültig "erlernt" zu sein. In *SHERLOOK* wird das letztendlich so erlernte Zeichen "*Normzeichen*" genannt, und erst dieses dient in weiteren Erkennungsläufen als Vergleichsvorlage. Die so festgestellten Normzeichen sind in der Zeichensatztabelle dann invers gekennzeichnet.

Nur der Vollständigkeit halber

Sehr schön ist die Möglichkeit, das Lernen auszuschalten (Korrektur manuell). Dies bedeutet, daß ein Bild bei hinreichend genauen und vollzähligen Normzeichen nicht mehr weiter auf unbekannte Zeichen "erlernt" werden soll. Die unbekannten Zeichen markiert das Programm dann mit einem bestimmten Symbol und gibt den Text entsprechend aus.

Daß so banale Funktionen wie Zeichensatz löschen, speichern, laden und Text löschen, zeigen, entfernen, anzeigen, speichern (ASCII und 1st_Word) möglich sind, bedarf fast kaum noch der Erwähnung. Auch ist eine knappe aber nützliche Online-Hilfe eingebaut.

Zu guter Letzt

Erinnern Sie sich noch? Es ist noch gar nicht so lange her, daß ich in dieser Zeitschrift einen Artikel zur Texterkennung schloß mit dem Hinweis, wir würden alsbald mit neuen Informationen zu diesem Thema zurückkehren. Und ein kleinwenig freut es mich, daß dies so bald geschehen durfte. *SHERLOOK* reiht sich in eine kleine Gruppe von Programmen ein, die sich in gar nicht mehr so langer Zukunft einen Markt erobern werden, der uns heute noch etwas utopisch erscheint. Bald werden Scanner so preisgünstig sein, daß sie uns geläufig sind wie Nadeldrucker. *SHERLOOK* ist ein Programm zur Texterkennung, mit dem professionell gearbeitet werden kann. Da es vollständig in Assembler geschrieben ist, legt es akzeptable Geschwindigkeiten vor und braucht einen Vergleich mit ähnlichen Programmen nicht zu scheuen. Der Einbau kleiner, zunächst unscheinbarer Funktionen zeigt, daß sich die Entwickler sehr ernsthaft mit der Problematik befaßt haben.

Dieter Kühner

Bezugsquelle:

H. Richter
-Distributor-Hagener Straße 65
5820 Gevelsberg
Telefon: 02332/2706

Z88 Ein Finite-Elemente-Programm

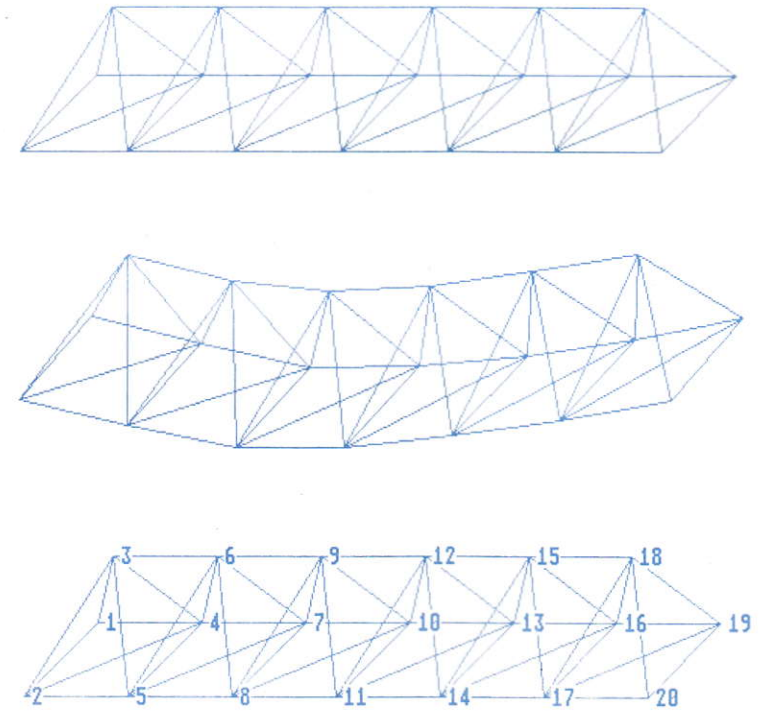
Ohne Computer wäre die Berechnung Finiter Elemente gar nicht denkbar, da der Rechenaufwand immens ist. Daher wurden schon vor Jahren Programme geschaffen, die diese Berechnungen übernahmen. Allerdings waren der Speicherbedarf und die Rechenzeit so groß, daß sich der Einsatz kleinerer Rechner nicht lohnte. Wer Finite Elemente berechnen wollte, der mußte schon einen Großrechner bemühen. Z88 ermöglicht nun die Berechnungen auch auf kleineren Rechnern. Was ein derartiges Programm leistet, soll im folgenden aufgezeigt werden.

Zuerst aber die Frage, die sicherlich viele LeserInnen interessiert: Was sind Finite Elemente? Finite Elemente sind dem Bereich der numerischen Mathematik zuzuordnen. Sie werden benötigt, um Näherungslösungen für diverse Probleme zu bestimmen.

Die eher klassischen Verfahren der Berechnung von Näherungslösungen numerischer Probleme auf Gittern warfen einige Probleme auf, da die Lösungen nur in ausgewählten Punkten - auf dem Gitter - berechnet wurden. Eine Näherungslösung auf einer Fläche (oder im Raum) war so nicht zu erhalten. Finite Elemente bieten nun den Vorteil, auf einzelnen Teilen einer Fläche Näherungswerte zu erhalten, womit eine "Flächen-"Lösung erreicht wird. Durch diese Art der Näherung ist es möglich, Funktionswerte an jedem Punkt einer Fläche nachträglich zu berechnen, ohne die Gesamtberechnung erneut durchführen zu müssen.

Finite-Elemente-Berechnungen können mit dem Programm Z88 sowohl im zwei- als auch im dreidimensionalen Raum durchgeführt werden.

Als Rechenbeispiel wurde eines der mitgelieferten Beispiele ausgewählt. Es handelt sich um einen Kranträger, der aus 54 Stäben besteht. Der Träger wird auf vier der Knoten aufgelegt, und an zwei Knoten wird eine Last angehängt. Ausgehend von dieser Situation berechnet Z88 die Verformung des Trägers. Aber zu dieser Berechnung wollen wir gleich wieder zurückkehren.



Z88 präsentiert sich mit zwei oder drei Disketten, je nachdem, für welche Version man sich entschieden hat. Für den ATARI stehen die ATARI-ST1- und die ATARI-ST4-Version zur Verfügung. Die erste Version ist für den kleinen ATARI ST mit 1 MB, die zweite für den großen mit 4 MB. Die kleine Version reicht für 50000 Elemente in der Gesamtsteifigkeitsmatrix, 2000 Freiheitsgrade, 1000 Knoten und 500 finite Elemente. Die große Version hat die achtfache Kapazität. Der Diskettenbetrieb ist grundsätzlich auch möglich, jedoch reicht die Diskettenkapazität nicht aus. Es kann ohne weiteres passieren, daß Dateien 5 MB groß werden. Daher ist eine Festplatte dringend empfohlen. Wer professioneller mit Z88 arbeiten möchte, kommt um die ATARI-ST4-Version nicht herum. Da die Rechenzeiten sehr groß werden - die Berechnung finiter Elemente ist sehr aufwendig -, sollte auch ein Coprozessor 68881 vorhanden sein. Die Ausgabe erfolgt auf einen Drucker, ein Plotter dürfte jedoch bessere Ergebnisse liefern.

Die Hardware-Voraussetzungen zeigen, daß Z88 nicht ein Programm für jeden sein kann, was aber auch sonst logisch ist. Neben den ATARI-Versionen existiert Z88 auch

```
     3   20   54   60    1
     1    3 0.0      2000.     0.0
     2    3 0.0      0.0       0.0
     3    3 1000.    1000.     2000.
     4    3 2000.    2000.     0.0
     5    3 2000.    0.0       0.0
     6    3 3000.    1000.     2000.
     7    3 4000.    2000.     0.0
     8    3 4000.    0.0       0.0
     9    3 5000.    1000.     2000.
    10    3 6000.    2000.     0.0
    11    3 6000.    0.0       0.0
    12    3 7000.    1000.     2000.
    13    3 8000.    2000.     0.0
    14    3 8000.    0.0       0.0
    15    3 9000.    1000.     2000.
    16    310000.    2000.     0.0
    17    310000.    0.0       0.0
    18    311000.    1000.     2000.
    19    312000.    2000.     0.0
    20    312000.    0.0       0.0
     1    4
     1    2
500.
     2    4
     4    5
500.        3    4
     7    8
500.        4    4
    10   11
500.        5    4
    13   14
500.        6    4
    16   17
500.        7    4
    19   20
500.        8    4
     1    4
500.        9    4
     2    5
500.       10    4
     4    7
500.       11    4
     5    8
500.       12    4
     7   10
500.       13    4
     8   11

...

    54    4
    17   19
500.        1   54 200000.   0.3
```

Tabelle 1: Z88I1.TXT

```
Z88 Kommando-Prozessor fuer Atari ST
von Dr.-Ing. Frank Rieg, Darmstadt 1988, V 5.0

     Hauptmenue

Editieren mit ---> TEMPUS
N(etzgenerieren mit Z88N
P(lotten --->
V(erformungen rechnen --->
S(pannungen rechnen
K(raefte rechnen
F(ILE88 starten zum Erzeugen/Loeschen von Files

? Hilfe = <?><Kommando>
X(it
```

```
   10
    1    2    2 0.
    1    3    2 0.
    2    1    2 0.
    2    3    2 0.
    7    3    1 -30000.
    8    3    1 -30000.
   19    1    2 0.
   19    3    2 0.
   20    2    2 0.
   20    3    2 0.
  100 0.000001
```

Tabelle 2: Z88I2.TXT

```
    0    0    0
```

Tabelle 3: Z88I3.TXT

noch für MS-DOS und OS/2, wobei auch hier die Leistung eines MS-DOS-Rechners bzw. der MS-DOS-Version von Z88 nicht für professionelle Ansprüche ausreicht.

Z88 umfaßt mehrere, in Fortran 77 geschrieben Programme. Das System wurde modular gestaltet, damit möglichst viel Speicher für die Berechnung freibleibt. Zu den Programmen gehören neben den nötigen Übersetzungsprogrammen ein Netzgenerator, ein Spannungs- und ein Knotenkraftprozessor. Zur Lösung der Gleichungen stehen drei Verfahren zur Verfügung: Cholesky, Gauß und Iteration. Ergänzt wird das System durch ein Plot- und ein Bildschirmausgabeprogramm. Auf den mitgelieferten Disketten befinden sich ferner Beispiele, deren Benutzung klar beschrieben wird. Damit ist eine erste Inbetriebnahme des Programms sofort (nach der Installation auf der Festplatte) möglich. Dem System fehlt lediglich ein Editor, was aber nicht problematisch ist, da man so die Möglichkeit hat, seinen Lieblingseditor zu benutzen.

Das Handbuch wird im DIN A4-Format geliefert und erläutert neben dem Programm auch die Elemente und die Beispiele. Die Beschreibungen sind zwar nicht sehr ausführlich, genügen aber, um mit dem Programm klarzukommen. Ein Stichwortverzeichnis fehlt leider. Bei einem Programm wie Z88 ist jedoch nicht damit zu rechnen, daß ein absoluter Laie Berechnungen vornehmen wird, so daß das Handbuch kein Manko darstellt.

Z88 ist kein GEM-Programm. Es stellt diesen Anspruch aber auch nicht. Da die Menüsteuerung klar gestaltet wurde, ist neben der Umstellung von der gewohnten GEM-Umgebung auf eine "herkömmliche" Benutzerführung kein Problem zu erwarten. Alle Programme tragen die Endung PRG, womit es für den ST so wirkt, als liege ein GEM-Programm vor. Die Folge davon ist, daß der Mauszeiger erscheint (jedoch nicht benutzt werden kann), wenn man die Maus bewegt. Ferner erscheinen die Warnmeldungen des TOS als Alertbox, was zusätzlich ein paar Schwierigkeiten - wie das Anklicken der Buttons ohne Mauszeiger - mit sich bringt. Die Schwierigkeiten sind jedoch äußerst gering. Während des Z88-Tests kam es zu keinen Problemen.

Das Bildschirmausgabeprogramm nutzt das GEM, jedoch wird nur die Auflösung 640 mal 400 Pixel ausgenutzt. Ein Betrieb auf einem Ganzseitenmonitor ist zwar möglich, jedoch wird die höhere Auflösung leider nicht ausgenutzt. Getestet wurde dies mit BigScreen. Ferner stört bei einem modernen Programm, daß keine Umlaute ("ü" statt "ue", ...) benutzt werden.

Die Ausgabe der Hardcopy erfolgt auf konventionellem Wege mittels Alternate+Help. Speziell für NEC-Drukker wird HCOPYNEC.PRG benutzt. Die Ausgabe der Hardcopy erwies sich als etwas problematisch, da unter den gegebenen Umständen ein (nicht mitgeliefertes) Programm zur Umleitung der Hardcopy in eine Datei für Abstürze sorgte. Hier fehlt sicherlich noch eine Ausgabemöglichkeit in eine Datei mit Standardformaten. Abgesehen von diesen kleineren Problemen zeigte sich Z88 als ein zuverlässiges Programm.

Sehr vorteilhaft für die Datenübermittlung ist die Ein- und Ausgabe von Daten über ASCII-Dateien. Somit liegen die Daten gleich in verwertbarer Form für Z88 und ein Druckprogramm vor.

Kommen wir nun zu einer Beispielrechnung. Um es gleich vorwegzunehmen:

Rechenzeiten wurden nicht getestet, da sie erwartungsgemäß (mit dem Laden und Schreiben der notwendigen Dateien) im Minuten-Bereich liegen und auch keine Vergleichsmöglichkeiten vorhanden waren. Da aber nur akzeptable Zeiten aufgetreten sind, kann man Z88 eigentlich nur ein Lob aussprechen.

Das Rechenbeispiel wird im Handbuch erläutert. Wie oben bereits beschrieben, handelt es sich um einen Kranträger, der aus 54 Stäben besteht und ein räumliches Fachwerk bildet. Der unverformte und der verformte Träger sind auf den Bildern zu sehen. Ferner zeigt eines der Bilder den Kranträger mit durchnumerierten Knoten. Der Träger wird in den Knoten 1, 2, 19 und 20 gelagert, auf die Knoten 7 und 8 wird eine Last von -30000 N gegeben. Der Träger hat eine Länge von 12 m, und als Werkstoff wurde Stahl [technische Daten: E=20000 N/m^2, nue=0.3 (nue ist ein griechischer Buchstabe, der sich leider nicht drucken läßt)]. Die Stäbe haben eine Querschnittsfläche von 500 mm^2. Aus den Grafiken läßt sich entnehmen, wie der Träger vor und nach der Verformung aussieht.

Die Eingabedatei Z88I1.TXT enthält in der ersten Zeile die Dimension der Struktur (3), die Anzahl der Knoten (20), die Anzahl der Elemente (54), die Anzahl der Freiheitsgrade (60) und ein Koordinaten-Flag (1) für Polar- bzw. Zylinderkoordinaten.

Ab der zweiten Zeile folgt dann die Knotenliste mit den jeweiligen Freiheitsgraden und Koordinaten. Nach der Knotenliste folgt die Elementliste mit Elementnummer, -typ und Knotennummern sowie einem Querschnittsparameter (je drei Zeilen). Es tauchen nur zwei Knotennummern auf, da als Elementtyp das Stabelement gewählt wurde. In der letzten Zeile werden die Elastizitätsgesetze aufgeführt. Hier handelt es sich um das E-Modul 200000 mit Querkontraktionszahl nue=0.3 für die Elemente 1 bis 54.

Die Eingabedatei Z88I2.TXT bestimmt nun die Randbedingungen. Aufgeführt ist in der ersten Zeile die Anzahl (10) und dann folgend die Liste. Ab der zweiten Zeile sind jeweils in einer Zeile die Knotennummer, der Freiheitsgrad und das Steuer-Flag (1 für Kraft, 2 für Verschiebung vorgeben), in der letzten Zeile die maximale Iterationszahl (100) und die Fehlerschranke (0.000001) angegeben.

```
   Z88PVDI Plotprogramm fuer Finite Elemente Programm Z88
       von Dr.-Ing Frank Rieg, Darmstadt 1989 V5.02

   Interface fuer Plotter (AUX:,..): Z8806.TXT
   Strukturfile (vom Typ Z88I1.TXT): Z88I1.TXT
   Laden des/der Files

   Unverformte oder Verformte Struktur plotten

   CRT- oder Plotter-Ausgabe
   Ansicht:   0 XY oder 1 XZ oder 2 YZ oder 3 3Dim
   No Labels oder Knoten labeln oder Elemente labeln oder Alles labeln

   Faktoren aendern                      Zustand der Spannungen

   Go - Start Plotten                    Xit - Ende Plotprogramm Z88PVDI

                        Eingestellte Faktoren

FACX:    1.00000        CX:   0.000000E+00      FUX:    100.000
FACY:    1.00000        CY:   0.000000E+00      FUY:    100.000
FACZ:    1.00000        CZ:   0.000000E+00      FUZ:    100.000
```

```
DIMENS  KNOTEN  ELEMEN  FREIHE  E-GESE  KFLAG
   3       20      54      60       1      0
KNOTEN  FG     X                 Y                Z
     1    3 0.00000000E+00 2.00000002E+03 0.00000000E+00
    2    3 0.00000000E+00 0.00000000E+00 0.00000000E+00
    3    3 1.00000001E+03 1.00000001E+03 2.00000002E+03
    4    3 2.00000002E+03 2.00000002E+03 0.00000000E+00
    5    3 2.00000002E+03 0.00000000E+00 0.00000000E+00
    6    3 3.00000011E+03 1.00000001E+03 2.00000002E+03
    7    3 4.00000005E+03 2.00000002E+03 0.00000000E+00
    8    3 4.00000005E+03 0.00000000E+00 0.00000000E+00
    9    3 5.00000000E+03 1.00000001E+03 2.00000002E+03
   10    3 6.00000023E+03 2.00000002E+03 0.00000000E+00
   11    3 6.00000023E+03 0.00000000E+00 0.00000000E+00
   12    3 6.99999988E+03 1.00000001E+03 2.00000002E+03
   13    3 8.00000011E+03 2.00000002E+03 0.00000000E+00
   14    3 8.00000011E+03 0.00000000E+00 0.00000000E+00
   15    3 8.99999976E+03 1.00000001E+03 2.00000002E+03
   16    3 1.00000001E+04 2.00000002E+03 0.00000000E+00
   17    3 1.00000001E+04 0.00000000E+00 0.00000000E+00
   18    3 1.09999999E+04 1.00000001E+03 2.00000002E+03
   19    3 1.20000004E+04 2.00000002E+03 0.00000000E+00
   20    3 1.20000004E+04 0.00000000E+00 0.00000000E+00

 ELEMT TYP    K1   K2   K3   ...   K20      QPARA
    1    4     1    2                       500.
    2    4     4    5                       500.
    3    4     7    8                       500.
    4    4    10   11                       500.
    5    4    13   14                       500.
    6    4    16   17                       500.
    7    4    19   20                       500.
    8    4     1    4                       500.
    9    4     2    5                       500.
   10    4     4    7                       500.
   11    4     5    8                       500.
   12    4     7   10                       500.
   13    4     8   11                       500.

...

   54    4    17   19                       500.

     VON  BIS   E-MODUL    NUE       INTORD
      1    54  2.000E+05  0.300         0
```

Tabelle 4: Z88O1.TXT

Die Eingabedatei Z88I3.TXT hat keinen Einfluß auf die Berechnung und enthält demgemäß keine interessanten Daten. Nach Durchlauf des Berechnungsprozesses erhält man die Ausgabedateien Z88O1.TXT und Z88O2.TXT. Die anderen Ausgabedateien (Spannungsberechnung und Knotenkräfte) sind hier wegen der Größe nicht abgebildet. Der Aufbau der Datei Z88O1.TXT ist analog zum Aufbau von Z88I1.TXT und wird daher nicht erneut erläutert.

Die Datei Z88O2.TXT enthält die resultierenden Verschiebungen aufgelistet nach Knoten in x-, y- und z-Richtung. Soweit das Beispiel.

VERSCHIEBUNGEN
****

KNOTEN	U(1)	U(2)	U(3)	...
1	-8.89430727D-01	0.00000000D+00	0.00000000D+00	
2	0.00000000D+00	-2.70673615D-01	0.00000000D+00	
3	2.79583333D+00	1.51748628D-01	-2.42278968D+00	
4	-7.10881443D-01	3.00984445D-01	-4.75756111D+00	
5	-2.82072375D-01	2.51761545D-01	-4.82541088D+00	
6	1.99583333D+00	2.65297840D-01	-6.78479376D+00	
7	-2.24456491D-01	5.49222788D-02	-8.65615841D+00	
8	-7.20204177D-02	3.05699380D-01	-8.65258330D+00	
9	3.95833333D-01	1.02474057D-01	-8.49628901D+00	
10	2.69844129D-01	-1.75388550D-01	-8.25197649D+00	
11	3.30155871D-01	-2.24611450D-01	-8.07697649D+00	
12	-8.04166667D-01	-1.52474057D-01	-7.25727542D+00	
13	4.72020418D-01	-5.05699380D-01	-6.34955612D+00	
14	6.24456491D-01	-5.54922279D-01	-6.05313124D+00	
15	-1.60416667D+00	-3.15297840D-01	-4.77002338D+00	
16	3.82072375D-01	-4.51761545D-01	-3.39889729D+00	
17	8.10881443D-01	-5.00984445D-01	-3.18104752D+00	
18	-2.00416667D+00	-2.01748628D-01	-1.63453288D+00	
19	0.00000000D+00	1.70673615D-01	0.00000000D+00	
20	8.89430727D-01	0.00000000D+00	0.00000000D+00	

Tabelle 5: Z88O2.TXT

Zusammenfassend läßt sich Z88 als ein zuverlässiges, preisgünstiges Finite-Elemente-Programm bezeichnen. Die Leistungen reichen aus, um - unter Bereitstellung der nötigen Hardware - auch auf kleineren Rechnern wie dem ATARI ST komplexe Berechnungen durchzuführen. Die Nicht-GEM-Umgebung ist klar gestaltet und läßt sich problemlos bedienen. Die Rechenzeit ist für ein Finite-Elemente-Programm als durchaus kurz zu bezeichnen.

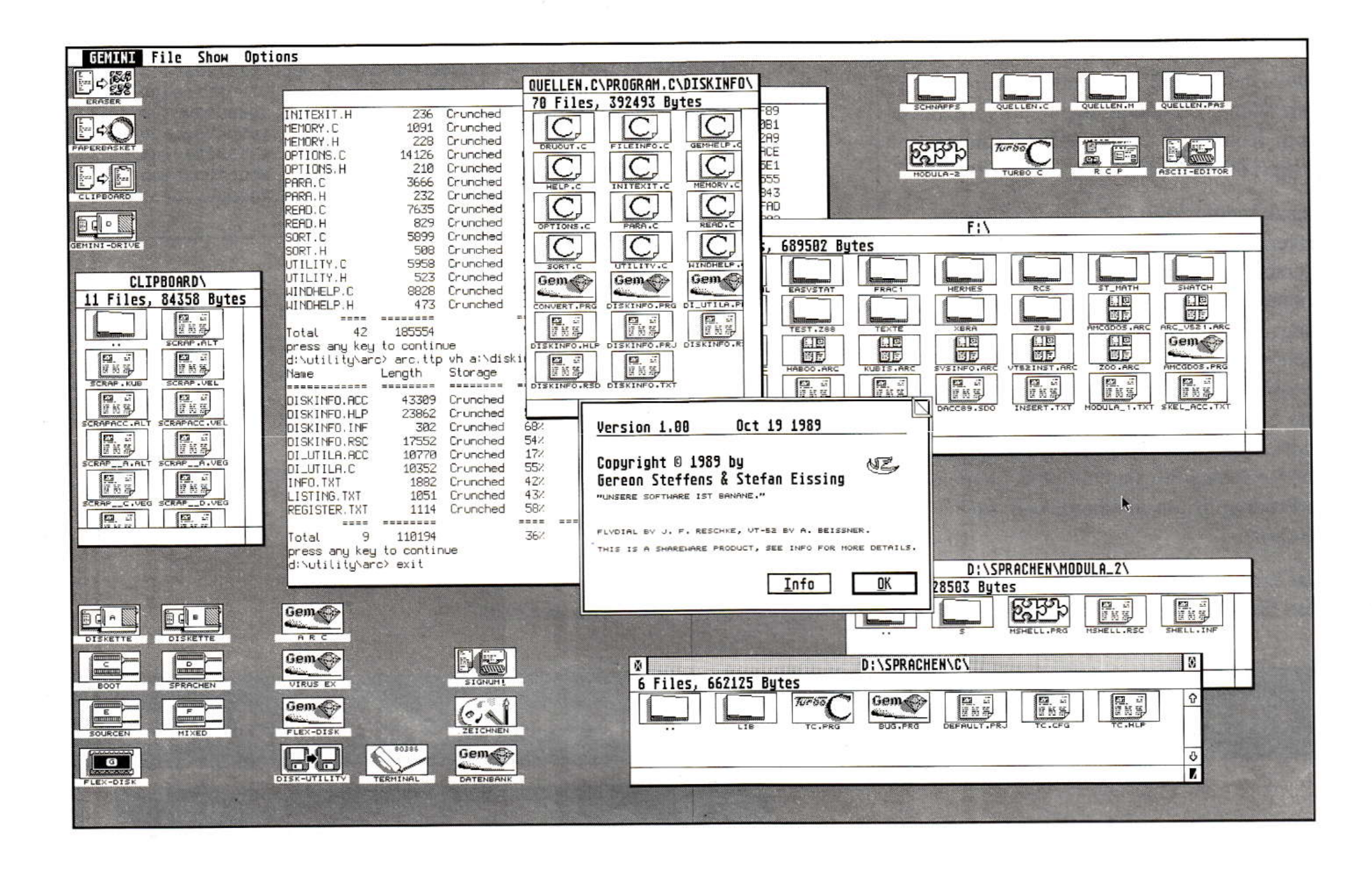
Das Programm kostet in der kleinen Version 198.- DM, in der größeren 498.- DM, wobei bei der größeren Fassung auch der Coprozessor 68881 unterstützt werden kann. Der Vollständigkeit halber seien auch die Preise der anderen Versionen genannt: MS DOS 498.- DM, OS/2 598.- DM. Das Handbuch kann für 48.- DM einzeln bestellt werden, eine Demo-Version ist nicht erhältlich.

Dietmar Rabich

Bezugsadresse:

HPS Gesellschaft für Entwicklung und Vertrieb von Soft- und Hardware mbH
Karlsbader Straße 10
6100 Darmstadt
Tel. (06151) 316132

Die GEMINI-Shell



Heute berichten wir über ein bemerkenswertes Programm aus dem Shareware-Bereich. GEMINI - seit Oktober 1989 erhältlich - ist eine der umfangreichsten und interessantesten Shells, die es im ST-Bereich gibt.

Das Programm stellt eine Vereinigung von *MUPFEL* und *VENUS* dar, wobei MUPFEL der klassische (Command-Line-Interpreter) und VENUS der grafische Teil der Shell ist. Die Vereinigung der Maus- und Tastaturbedienung ist sehr gut gelungen. Es geht sogar soweit, daß TOS-Programme in Fenstern ablaufen! Alle Kommandos können über Mausklick oder Tastatur gegeben werden. Selbst die Dialog- und Alert-Boxen ermöglichen eine vollständige Tastatursteuerung. Bei den Dialog- und Alert-Boxen ("Fliegende Dialoge") handelt es sich um "selbstgestrickte", die sich frei auf dem Bildschirm positionieren lassen.

Auf dem Desktop sind die üblichen Laufwerkssymbole zu finden. Zusätzlich dazu erreicht man das Clipboard, das GEMINI-Laufwerk, einen Papierkorb, aus dem man Dateien wieder herausholen kann, und einen Reißwolf, der Dateien endgültig löscht.

Dateien, Ordner und Programmsymbole können auf dem Desktop abgelegt werden, womit Programme schnell und ohne Sucherei gestartet werden können. Dateien werden an Programme übergeben, indem das Dateisymbol auf das entsprechende Programmsymbol geschoben wird. Das betreffende Programm startet umgehend und lädt - sofern es dazu in der Lage ist - die Datei nach.

Die Symbole oder Icons sind frei wählbar, auch für bestimmte Dateien oder Dateigruppen. Ebensolche Wahlmöglichkeiten bestehen bei den Laufwerkssymbolen.

Die Fenster sind wie üblich maus- und tastatursteuerbar. Mit einem Druck auf eine Taste - etwa "E" - werden sofort alle Programme angezeigt, die mit diesem Buchstaben beginnen. Horizontales Scrollen im Fenster ist nicht nötig, da immer nur soviele Icons in der Horizontalen angezeigt werden, wie das Fenster breit ist.

Im "VENUS"-Menü findet sich, wie es sich bei GEM-Programmen gehört, ein Eintrag, mit dem eine Programminformation abgerufen werden kann.

Das "File"-Menü ermöglicht das Öffnen von Dateien, das Abrufen von Informationen zu Disketten, Dateien oder Ordnern. Ebenso lassen sich neue Ordner anlegen, Fenster schließen (analog zur Close-Box im Fenster), Fenster völlig schließen, die Reihenfolge der Fenster ändern und natürlich das Programm verlassen.

Die Darstellung im Fenster läßt sich im "Show"-Menü einstellen. Hierbei hat man die Text- oder die Icon-Darstellung zur Auswahl, wobei Icons in zwei Größen vorhanden sind. Die Daten in den Fenstern können Sie nach Namen, Datum, Größe, Typ oder gewählten Icons sortieren. Ferner besteht die Möglichkeit, die Daten unsortiert darzustellen. Mit Wildcards kann man für jedes Fenster festlegen, welche Dateien angezeigt werden sollen.

Das "Options"-Menü ermöglicht die Wahl der Disk- und File-Icons. Verschiedene Dateitypen können - analog dem Desktop - Programmen zugeordnet werden. Über "Display" wird die Darstellung in Fenstern beeinflußt. So können beispielsweise nur die Dateinamen mit Datum angezeigt werden, Größe und Zeit werden unterdrückt.

Die Mupfel (Text-Shell) wird, wie auch die dazugehörige Einstellmöglichkeit, über das "Options"-Menü erreicht. Bemerkenswert ist, daß man unterschiedliche Fonts wählen kann. Somit ist auch die Darstellung von 25 x 80 Zeichen im Fenster ohne weiteres möglich.

Zu guter Letzt ist auch der Blitter an- und abschaltbar, und es können die allgemeinen Optionen eingestellt werden. Natürlich besteht auch die Möglichkeit, die gewählten Einstellungen abzuspeichern.

Die Kommandos von Mupfel lehnen sich an die von UNIX bekannten an. Dazu gehören auch Batch-Dateien, I/O-Redirection, Environment, History-Funktion und Aliase. Die UNIX-Wildcards (mächtiger als die des GEMDOS) werden sowohl von der Mupfel als auch von der Venus unterstützt.

Soviel zur Beschreibung von GEMINI. Sicherlich kann hier nicht jedes Feature von GEMINI erwähnt werden, denn dazu ist die Shell viel zu mächtig.

Die tägliche Arbeit mit dem ST wird durch GEMINI wesentlich erleichtert. Nach wenigen Stunden sehnt man sich nicht mehr zum normalen Desktop zurück, denn kaum jemand möchte eine Arbeitsumgebung wie GEMINI vermissen. Wie die Autoren in ihrem README-File schreiben, ist die Entwicklung der Shell noch nicht abgeschlossen. Für die Zukunft haben sie sich noch Pipes, eine Script-Sprache, Tastatur-Shortcuts für den Programmstart und eine Funktion zum Aufräumen des Desktops vorgenommen.

GEMINI ist ein (mit Turbo C geschriebenes) Programm, welches auf dem ST neue Maßstäbe setzt. Die bekannten GEM-Konventionen und Richtlinien wurden eingehalten. Somit kommt GEMINI durchaus eine Vorbildfunktion zu.

Die Shell kann auf jedem ST mit mindestens TOS 1.2 betrieben werden. GDOS wird benötigt, wenn andere als die Standardzeichensätze für Mupfel Verwendung finden sollen. Ein Megabyte Speicher sollte für das mit Resourcefile knapp über 200 kB-große Programm schon vorhanden sein. Wer eigene Icons ergänzen möchte, ist ferner auf ein Resource-Construction-Set mit Icon-Editor angewiesen.

Das Programm ist ein Shareware-Programm. Es darf jedoch von keinem Public Domain-Vertrieb auf Diskette vertrieben werden, da es die AnwenderInnen auf kostenlosem Weg erreichen soll.

Die Entwickler (*Gereon Steffens, Elsterweg 8, 5000 Köln 90 und Stefan Eissing, Dorfbauernschaft 7, 4419 Laer-Holthausen*) versenden GEMINI gegen Zusendung einer 3 1/2"-Diskette und eines adressierten und frankierten Rückumschlags. Ferner ist GEMINI über die *Mailboxen Maus Münster* (0251-80386) und *Maus Bonn* (0228-254020) abrufbar.

Zusammenfassend soll hier betont werden, daß GEMINI ein Programm ist, daß man sich nicht entgehen lassen sollte. Schließlich ist bei der Nutzung ja nur ein Shareware-Beitrag zu leisten. GEMINI wird bestimmt viele Freundinnen und Freunde finden.

Dietmar Rabich

Digitalisieren in vier Graustufen

auf dem Monochromschirm

Die Idee zu diesem Programm entstand, als ich etwas enttäuscht die Qualität der digitalisierten Bilder des Easytizers im Monochrombetrieb sah. In diesem Modus gab es nur schwarze oder weiße Flächen und bestenfalls ein sehr grobes Raster; arbeitete man in der mittleren Auflösung und ließ es in ein hochauflösendes Bild umrechnen (für Ausdrucke unumgänglich), war das Ergebnis auch nicht berauschend. Da der Digitizer in vier Graustufen bis zu einer Auflösung von ca. 760*500 arbeitet, entstand bald die Idee, das für den Monochrommonitor zu nutzen. Intern legt das Programm ein Bild mit der Auflösung 640*400 und zwei Planes ab. Dieses Bild ist übrigens speicher- (64 kByte) und ladbar. Um gleich eine ungefähre Vorstellung zu haben, was nun eigentlich im Speicher ist, wird ein zweites Bild digitalisiert (s/w) und sofort angezeigt. Dies geschieht immer, egal ob einmal oder fortlaufend digitalisiert wird. Über einen Menüpunkt aus dem Menü "Rechnen" wird nun das vierfarbige Bild in ein monochromes umgerechnet.



Es stehen zur Zeit drei Möglichkeiten zur Verfügung:

1. Umrechnen in ein definiertes Muster
2. Umrechnen über eine zufällige Verteilung
3. wie zuvor, nur werden horizontal zufällige Aneinanderreihungen von gleichfarbigen Pixeln vermieden.

Diese Umrechnungsroutinen sind in 'C' geschrieben und dürften ausreichend schnell sein. Beim Versuch, diese Routinen in BASIC zu schreiben, dauerte eine Umrechnung fast 10 Minuten.

Im Menü *Parameter* verbirgt sich die Auswahl der Muster bzw. die *Helligkeit* für die Zufallsmuster. Bei der Musterauswahl ist zu beachten, daß jenes Muster gilt, das nach dem Invertieren mit dem 'OK'-Zeichen versehen ist. Man hat hier die Möglichkeit, auch die Farben Schwarz und Weiß zu ändern, während das bei den Zufallsmustern nicht möglich ist.

Beim Menüpunkt *Zufall* können nur drei Standardeinstellungen für Hell- bzw. Dunkelgrau gewählt werden. Neben den Buttons steht eine Prozentzahl, die angibt, wie hell bzw. dunkel der Grauton ist.

Die Menüpunkte aus dem Menü *Rechnen* können natürlich mehrmals angewählt werden, die große Bit-Map bleibt bis zum nächsten Digitalisieren erhalten. Das letzte Menü 'Invertieren' dient vor allem diversen Snapshot-Programmen (*Snapshot* von GST, *Scrcop* von Application Systems), da das Programm mit invertiertem Bildschirm arbeitet und so alle auf diese Art abgespeicherten Bilder negativ wären. Nebenbei wird gleichzeitig die Menüleiste abgeschaltet, damit das gesamte Bild zur Verfügung steht. Mit der Taste 'I' wird alles wieder normal. Über den Menüpunkt *Sichern* kann, wie oben erwähnt, das gesamte Bit-Map oder der

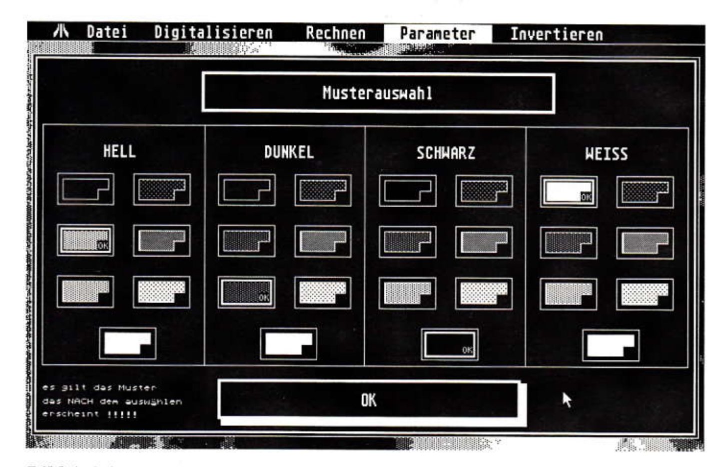


Bild 1: Die Musterauswahl des Programms

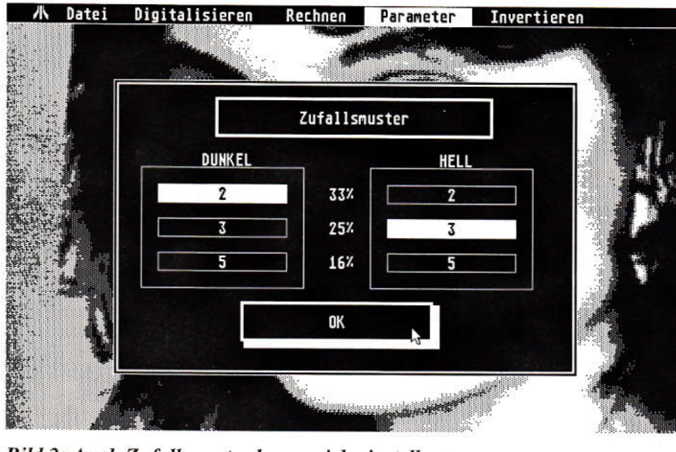


Bild 2: Auch Zufallsmuster lassen sich einstellen.

Bildschirm im normalen Degas-Format (32034 Byte, Endung .PI3) gespeichert werden. Geladen können nur Bit-Maps werden, die jedoch erst nach Anwahl eines Punktes aus dem Menü 'Rechnen' auf dem Bildschirm dargestellt werden.

Einige Tips für bessere Bilder

Der Easytizer ist sehr rotempfindlich, man sollte deshalb darauf achten, kein Bild mit großen grellroten Flächen als Vorlage zu nehmen (dies gilt natürlich nur, wenn man den Easytizer mit der gesamten Farbinformation füttert. Verwendet man z.B. eine Schwarzweißkamera, ist das egal). Obwohl der Digitizer sehr schnell ist, sollte man ein gutes, am besten ein digitales Standbild zur Verfügung haben. Sehr gute Ergebnisse erzielt man, wenn man mit einer Videokamera ein Foto etc. aufnimmt und sofort zum Rechner sendet. Legt man die Kamera auf einen Tisch und stellt das Bild 7-20 cm entfernt auf, erreicht man fast dasselbe wie mit einem teuren Reprostativ. Man muß nur für genug Licht sorgen; aber Vorsicht, auch zuviel Licht beeinträchtigt die Qualität. Sollte die Kamera über einen Backlight-Schalter verfügen, kann man damit den Kontrast verändern.

Spezielles über das Programm

Die Digitalisierroutinen sind aus der Zeitschrift ST-Computer, Ausgabe 9/87, entnommen. Die Umrechnungen bestehen eigentlich aus einfachen Bitverschiebe- und -kopierbefehlen. Das größte Problem war die Einbindung in das GFA-Programm. In der Anleitung des BASICs und in diversen Zeitschriften ist zwar wunderbar beschrieben, wie man C-Programme mittels des Befehls C: aufruft; wie man jedoch einen C-Compiler dazu bewegt,

```
 1:  /*  PHOTOBOX  */
 2:  /*  R.WIESLER 1989  */
 3:  /*  photo_10 */
 4:  /*  (c) MAXON Computer GmbH
 5:  #include <stdlib.h>
 6:
 7:  /*#include <time.h>*/
 8:
 9:  void zufall_1(void);
10:  void zufall_2(void);
11:  void muster(void);
12:
13:  typedef struct {
14:   unsigned int  was;
15:   unsigned int  *adresse;
16:   unsigned int  h1;
17:   unsigned int  h2;
18:   unsigned int  h3;
19:   unsigned int  h4;
20:   unsigned int  d1;
21:   unsigned int  d2;
22:   unsigned int  d3;
23:   unsigned int  d4;
24:   unsigned int  s1;
25:   unsigned int  s2;
26:   unsigned int  s3;
27:   unsigned int  s4;
28:   unsigned int  w1;
29:   unsigned int  w2;
30:   unsigned int  w3;
31:   unsigned int  w4;
32:   unsigned int z_hell;
33:   unsigned int z_dunkel;
34:   unsigned int *schirm;
35:   unsigned int zufallszahl;
36:  }*ZEIGER;
37:
38:  ZEIGER adr;
39:
40:  void cdecl photobox(ZEIGER adr)
41:  {
42:   switch(adr->was)
43:   {
44:   case 1: muster();        break;
45:   case 2: zufall_1();      break;
46:   case 3: zufall_2();      break;
47:   }
48:  }
49:
50:  /* zufall_1  */
51:
52:  void zufall_1()
53:  {
54:   unsigned int wa,wb,c,potenz;
55:   unsigned int *k,*l,k1,l1,*retour;
56:   register unsigned int i,j,bit;
57:   int wert_0;
58:   srand(adr->zufallszahl % 37);
59:   for (i=0;i < 400;i++)
60:          {
61:          for (j=0;j < 80;j+=2)
62:                 {
63:                 k=(unsigned int *) adr->adresse;
64:                 adr->adresse+=2;
```

→

einen Code zu liefern, der auf diese Art und Weise aufrufbar ist, ist nirgends zu finden. Um anderen Anwendern unzählige Abstürze zu ersparen, einige Tips, wie ich das Problem löste.

Um ein C-Programm in BASIC einzubinden, muß es frei im Speicher verschiebbar sein. Aus diesem Grund dürfen keine absoluten Adressen und keine Relozierdaten vorhanden sein. Bei den Compileroptionen ist bei Turbo-C jedoch kein Punkt vorhanden, der dies gewährleistet. Man schreibt also sein Programm, darf aber das Wort *main* nicht verwenden.

```
void cdecl photobox(ZEIGER adr)
{
      switch(adr->was)
      {
      case 1: muster();      break;
      case 2: zufall_1();    break;
      case 3: zufall_2();    break;
      }
}
```

Das Programm sieht also aus wie ein Unterprogramm. Für die Datenübergabe verwendet man am besten Strukturen und übergibt nur die Adresse der Struktur.

```
typedef struct {
      unsigned int was;
      unsigned int *adresse;
unsigned int h1;
   ...        unsigned int *schirm;
unsigned int zufallszahl;
}*ZEIGER;
```

Strukturen sollte man übrigens viel öfter benutzen, sie sind eine tolle Sache. Mit Variablen sollte man sehr sparsam umgehen und sie möglichst als *register* definieren. Danach compiliert man das Ganze mit den Standardeinstellungen. Für das Linken muß man sich eine spezielle Projektdatei erstellen. Man nimmt die *STANDARD.PRJ* und löscht die Zeile mit *TCSTART.O*. Dadurch wird verhindert, daß der Linker den Programmkopf und die Reloziertabelle dazuhängt. Das entstandene verstümmelte Programm sieht man sich am besten mittels eines Disassembler an, ob ja keine absoluten Adressen vorhanden sind. Ist alles in Ordnung, kann man das Programm ins GFA-BASIC einbinden. Dies geht am einfachsten mit dem Befehl *INLINE*.

```
INLINE c_prog%,2000
```

Die Datenübergabe kann mittels *POKEs* erfolgen (wie im Programm *PHOTOBOX*) oder etwas eleganter über Felder.

```
65:                     l=k+1;
66:                     wert_0=0;
67:                     potenz=1;
68:                     k1=*k;
69:                     l1=*l;
70:                     for(bit=1;bit<17;bit++)
71:                             {
72:                             wa=k1 & 1;
73:                             wb=l1 & 1;
74:                             c=0;
75:                             if (wa == 1)
76:                                     {
77:                                     if (wb == 1)
78:                                             c=potenz;
79:                                     else
80:                                             {
81:                                             if(rand()<adr->z_hell)
82:                                             c=potenz;
83:                                             }
84:                                     }
85:                             else
86:                                     {
87:                                     if (wb == 1)
88:                                             {
89:                                             if(rand()<adr->z_dunkel)
90:                                             c=potenz;
91:                                             }
92:                                     }
93:                             wert_0+=c;
94:                             potenz*=2;
95:                             k1 = k1 >> 1;
96:                             l1 = l1 >> 1;
97:                     }
98:
99:                     retour=(unsigned int *)adr->schirm + i*40 +(j/2);
100:                    *retour=wert_0;
101:                    }
102:                    }
103:    }
104:
105:
106:    void zufall_2()
107:    {
108:     unsigned int wa,wb,c,potenz;
109:     unsigned int *k,*l,k1,l1,*retour;
110:     register unsigned int i,j,bit;
111:     int wert_0,mark1,mark2;
112:     srand(adr->zufallszahl % 37);
113:     mark1=0;
114:     mark2=0;
115:
116:     for (i=0;i < 400;i++)
117:             {
118:             for (j=0;j < 80;j+=2)
119:                     {
120:                     k=(unsigned int *) adr->adresse;
121:                     adr->adresse+=2;
122:                     l=k+1;
123:                     wert_0=0;
124:                     potenz=1;
125:                     k1=*k;
126:                     l1=*l;
127:                     for(bit=1;bit<17;bit++)
128:                             {
129:                             wa=k1 & 1;
130:                             wb=l1 & 1;
131:                             c=0;
132:                             if (wa == 1)
133:                                     {
134:                                     if (wb == 1)
135:                                             c=potenz;
136:                                     else
137:                                             {
138:                                             if(rand()<adr->z_hell)
139:                                                     {
140:                                       if (mark1 == 1)
141:                                                     {
142:                                             c=potenz;
143:                                            mark1 = 0;
144:                                                     }
145:                                                else
146:                                            mark1 = 1;
147:                                                     }
148:                                             }
149:                                     }
```

→

```
DPOKE c_dat%+40,dunkel%
LPOKE c_dat%+42,schirm_adresse%
DPOKE c_dat%+46,zufall_startzahl%
~C:c_prog%(L:c_dat%) 'c_dat%
ist die Adresse der Struktur
'besser und kürzer wäre das Verwenden
eines Feldes
```

Eine andere Möglichkeit ist, das disassemblierte Programm in den GFA-Assembler zu laden. Hier läßt sich vielleicht noch etwas ändern oder optimieren (wenn man kann). Danach startet man vom Assembler aus das GFA-BASIC, lädt sein BASIC-Programm und stellt den Cursor auf den Befehl *INLINE*. Wenn man jetzt die HELP-Taste drückt, erscheint am oberen Bildschirmrand eine Menüzeile, die neben den bekannten Punkten den Punkt *ASS* enthält. Nach dessen Anwahl ist man wieder im Assembler. Nun kann man das Unterprogramm assemblieren, es wird frei verschiebbar in den *INLINE*-Befehl geschrieben. Diese Vorgehensweise ist auch in der Anleitung für den GFA-Assembler beschrieben; wenn man sie oft genug liest, findet man sie. Nach dem Verlassen des Assemblers ist man wieder im BASIC und kann alles sichern. Man sollte aus Sicherheitsgründen auch den *INLINE* abspeichern. Diese Möglichkeit des Zusammenspiels bietet natürlich nur der GFA-Assembler in Verbindung mit dem GFA-BASIC 3.0. Das GFA-BASIC-Programm ist wegen des Umfangs (ca. 1000 Zeilen) nur auf der Programmierpraxisdiskette *GFA-BASIC 1* zu finden.

Reinhard Wiesler

```
150:                    else
151:                            {
152:                            if (wb == 1)
153:                                    {
154:                                    if(rand ()<adr->z_dunkel)
155:                                            {
156:                                            c=potenz;
157:                                            mark2 = 0;
158:                                            }
159:                                    else
160:                                            {
161:                                            if (mark2 == 1)
162:                                                    {
163:                                                    c=potenz;
164:                                                    }
165:                                            else
166:                                                    {
167:                                                    mark2 = 1;
168:                                                    }
169:                                            }
170:                                    }
171:                            }
172:                    wert_0+=c;
173:                    potenz*=2;
174:                    k1 = k1 >> 1;
175:                    l1 = l1 >> 1;
176:            }
177:
178:            retour=(unsigned int *)adr->schirm + i*40 + (j/2);
179:            *retour=wert_0;
180:            }
181:            }
182:
183: }
184:
185:
186: /* muster  */
187:
188: void muster()
189: {
190:  unsigned int *k,*l,*retour;
191:  int wert_0;
192:  int i,j;
193:  int schwarz,weiss,hell,dunkel;
194:  int *schirm;
195:
196:
197:  for (i=0;i < 400;i++)
198:          {
199:          for (j=0;j < 80;j+=2)
200:                  {
201:                  k=(unsigned int *) adr->adresse;
202:                  adr->adresse+=2;
203:                  l=k+1;
204:                  schwarz=*k & *l;
205:                  weiss=  (~ *k) & (~ *l);
206:                  hell=   *l & (~ *k);
207:                  dunkel= *k & (~ *l);
208:
209:                  if (i % 4 == 0)
210:                          {
211:                          dunkel&=adr->d1;
212:                          hell&=adr->h1;
213:                          schwarz&=adr->s1;
214:                          weiss&=adr->w1;
215:                          }
216:                  else
217:                          {
218:                          if (i % 4 == 1)
219:                                  {
220:                                  dunkel&=adr->d2;
221:                                  hell&=adr->h2;
222:                                  schwarz&=adr->s2;
223:                                  weiss&=adr->w2;
224:                                  }
225:                          else
226:                                  {
227:                                  if (i % 4 == 2)
228:                                          {
229:                                          dunkel&=adr->d3;
230:                                          hell&=adr->h3;
231:                                          schwarz&=adr->s3;
232:                                          weiss&=adr->w3;
233:                                          }
234:                                  else
```

→

```
235:                    {
236:                    dunkel&
                        =adr->d4;
237:                    hell&=
                        adr->h4;
238:                    schwarz&
                        =adr->s4;
239:                    weiss&
                        =adr->w4;
240:                    }
241:
242:           }
```

```
243:          }
244:          wert_0=schwarz | weiss;
245:          wert_0|=hell;
246:          wert_0|=dunkel;
247:          retour=(unsigned int *)adr
              >schirm + i*40 +(j/2);
248:          *retour=wert_0;
249:          }
250:          }
251:  }
```

Die Umrechnungsroutinen in C

Der Calamus-Font-Editor

Die ersten Schritte zum Erfolg

Da liegt er, der neu entstandene Font-Editor für Calamus. Jetzt nur die Diskette mit zittrigen Fingern ins Laufwerk geschoben, und aus dem Reich der Beschränkung in das Reich der Freiheit... Augenblick, wo war doch die Betriebsanleitung...?

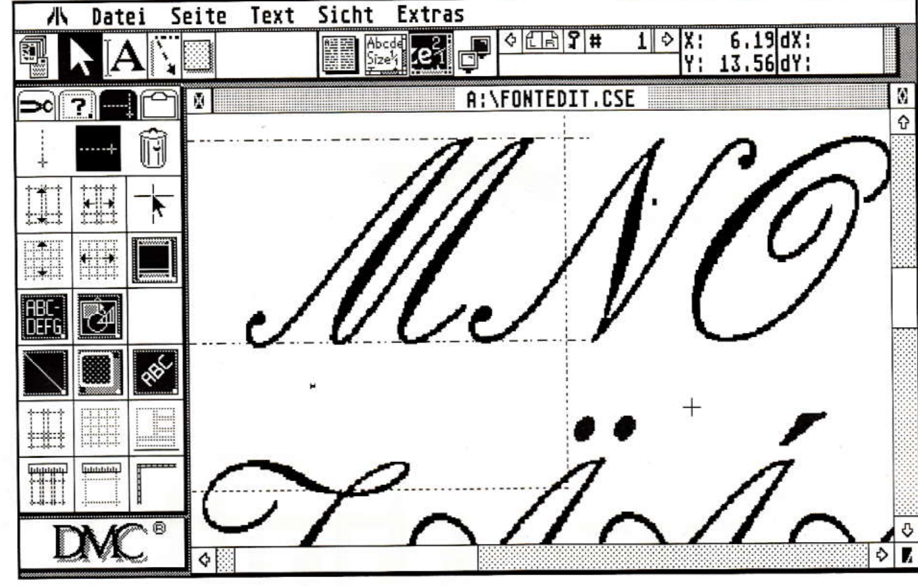


Abb.: 1

So wird es den meisten gehen, die die Möglichkeiten ihres Calamus mit dem Font-Editor erweitern wollen durch Eigenschöpfungen. Ausgangspunkt für die meisten ist wohl der Wunsch, gescannte Schriften in die hohe Calamus-Font-Qualität zu übertragen. Nach einigen Umwegen und Sackgassen kann ich einige Tips dazu beisteuern: Wenn Sie Ihre Vorlage scannen, achten Sie auf Rechtwinkligkeit! Speichern Sie möglichst in einem der Calamus-Bild-Formate. Da der Font-Editor ja ein Accessory ist, brauchen Sie ein Malprogramm, das Accessories zuläßt - oder Sie nehmen gleich Calamus. Ich habe mir zur Orientierung einen Linienrahmen gebaut, der quadratisch ist und die Verhältnisse des Font-Editors hat (Oberkante Großbuchstaben und Schriftlinie). Diesen Rahmen sollten Sie sich auch anfertigen aus dem Linien-Menü. Sie können ihn über jeden Buchstaben schieben, bis der Stand stimmt, und dann tritt die Kamera in Aktion, wenn Sie vorher aus dem Rahmen- in den Textmodus des Calamus gewechselt haben, denn sonst kann es passieren, daß sie während der Arbeit im Font-Editor im Hintergrund Rahmen auf- und zuziehen oder daß Ihr Cursor verschwindet...

Nach dem Start

Sie haben also Ihren Font-Editor gestartet, den Namen der Schrift und verschiedene Angaben eingetragen (Abb. 2) - nur die erste Zeile kann später ediert werden, damit der Urheber nicht entfernt werden kann - das richtige Buchstabenfeld (Abb. 3) angeklickt und die Kamera aktiviert. Klicken Sie auf Ausschneiden (Abb. 4) und führen Sie den Fadenkreuz-Cursor entlang Ihres erstellten Quadrats. Je größer die Wiedergabe auf dem Bildschirm, desto besser die Wiedergabe im Editor.

Wenn Sie die linke Maustaste loslassen, sehen Sie nach einer Wartezeit - nicht den ausgeschnittenen Bereich, sondern die Aufforderung "Ausschneiden". Das kann schon irritieren. Man kann aber jederzeit die oben vorwitzig über den Rand schauenden Icons anklicken (viertes von links)

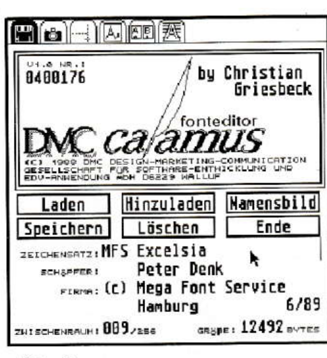


Abb.: 2

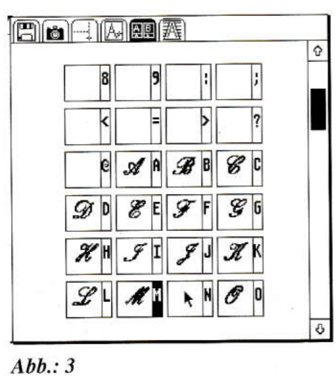


Abb.: 3

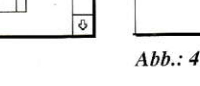


Abb.: 4

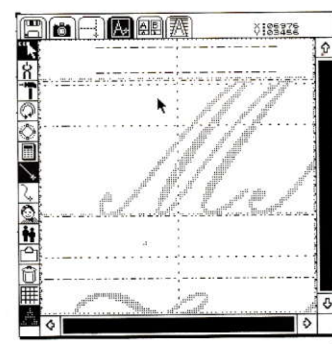


Abb.: 5

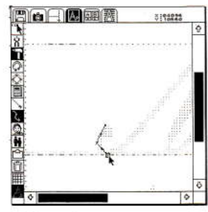

Abb.: 6


Abb.: 7

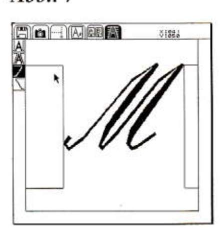
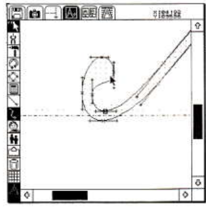
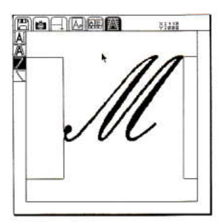
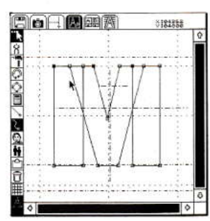
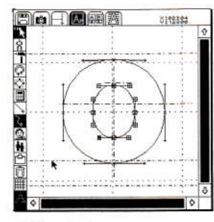

Abb.: 8


Abb.: 9


Abb.: 10


Abb.: 11


Abb.: 12


Abb.: 13

und man sieht endlich den ausgeschnittenen Buchstaben (Abb. 5). Jetzt kommt die erste Enttäuschung: Der Ausschnitt ist nicht verschiebbar. Wie gut, daß der Orientierungsrahmen, den man selbst gebaut hat, verschiebbar ist...

Schon haben Sie die Lupe aktiviert und das Kurven-Icon angeklickt und setzen die ersten Punkte, die durch Linien verbunden werden (Abb. 6). Es sieht ja noch ein wenig eckig aus (Abb. 7), aber man sucht sich markante Punkte (am einfachsten die waagerechten und senkrechten Scheitelpunkte jeder Kurve - Regel 1), bis der Buchstabe voll umkreist ist. Der letzte Punkt wird über den ersten gesetzt. Wenn Sie jetzt das Icon ganz rechts aktivieren, sehen Sie den Buchstaben schwarz (Abb. 8) und dazu die Kerning-Information (ein Kapitel für sich). Zu Anfang tut's das einfache Block- Kerning, die Feinheiten kommen später.

Entfaltung...

Mit dem dritten Icon von links kommen Sie in den Bereich der Hilfslinien. Hier können Sie (auch schräg) anordnen, was Sie an Hilfe brauchen. Es wird mit abgespeichert. Der nächste Schritt in der Erstellung der neuen Schrift ist das "Entfalten" der Tangenten, damit die Ecken entfernt werden. Man klickt auf den kleinen Querstrich in der Mitte des Verbindungsstriches zweier Punkte und zieht ihn bei gedrückter linker Taste zur Seite. Er verwandelt sich in ein Quadrat und eine Tangente, und der Verbindungsstrich krümmt sich... wenn Sie das zweite Feld unter der Mülltonne - ja, das leere Feld (!) - angeklickt haben. Sonst sehen Sie keine Tangente, und die Hilfspunkte sagen Ihnen nichts. Ist alles "entgratet", sieht man den fertigen Buchstaben in seiner ganzen Schönheit (Abb. 11).

Das Prinzip der Kurvenveränderung erklärt sich beim Arbeiten. Man kann sich die mitgelieferten Schriftenauch mal mit den Tangenten ansehen, um seine Schlüsse zu ziehen (Abb. 13). Dann wird man entdecken, daß man nicht alles immer wieder machen muß: Man kann Teile von Buchstaben kopieren und übereinanderschieben. Schieben? Ja, in der Anleitung ist es nicht ausdrücklich erwähnt, aber man kann bei gedrückter Alternate-Taste Pfade aktivieren (Abb. 13), die dann gelöscht oder verschoben oder kopiert werden können. Wenn man alles aktivieren will, kann man im Taschenrechner "select all" und danach "cancel" aktivieren.

...und Tangenten

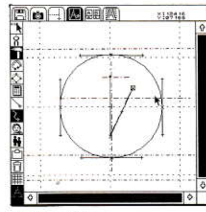
In unserem Beispiel habe ich die innere Kontur eines "O" aktiviert und gelöscht (Abb.14). Jetzt will ich sie neuzeichnen. Man braucht dabei gar nicht so viele Punktewie beim Vorbild: Für einen Kreis oder eine Ellipse genügen meist drei Punkte, wenn man sauber arbeitet... Hier sehen Sie den Zustand nach dem dritten Punkt (Abb. 15). Ich habe ihn noch nicht über den ersten geschoben, damit mehr Übersicht herrscht. Es ist besser, erst die Tangenten zu entfalten (Abb. 16 u. 17) und dann den Kreis zu schließen (Abb. 18). Im Vergleich der Abb. 20 und 21 können Sie sehen, was passiert, wenn man die Hilfspunkte nach innen schiebt. In der äußeren Kontur sehen Sie, was man vermeiden sollte: In der Tangente ist ein Knick, wo sie den Kreis berührt. Hier hat auch die Kurve einen (mit Übung) sichtbaren Knick. Deshalb ist unsere zweite Regel: Tangenten dürfen keinen Knick haben (außer, die Linie soll einen bekommen).

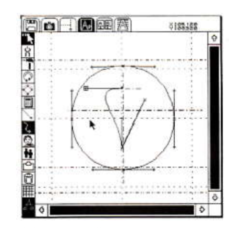
Jetzt haben wir schon gescannte Vorlagen sklavisch nachgemalt und aus eigenem inneren Erleben freie Kurven konstruiert. Der nächste Schritt ist das sog. Kerning (Festlegung, was der Kern des Buchstabens ist und wieviel Abstand er zum nächsten und vorhergehenden hat). Hier ist Calamus wirklich einzigartig, weil in der Höhe sieben verschiedene Kerning-Informationen links und rechts für jeden Buchstaben gespeichert werden. Aktivieren Sie einmal die Funktion Kerning optimiert (drittes Icon von obenauf der Kerning-Seite), dann sehen Sie die Kontur des Buchstabens von den sieben Kerning-Stufen umflossen. Man muß jetzt Phantasie entwickeln und sich vorstellen, was passiert, wenn bestimmte Buchstaben aufeinandertreffen...

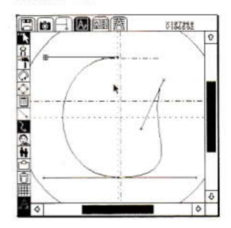
Orientierung

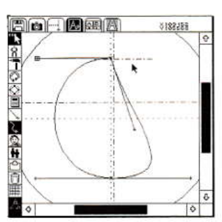
Eine große Hilfe ist auch hier das aufmerksame Betrachten der mitgelieferten Schriften. Es hat sich zum Beispiel bewährt, bei senkrechten Konturen von Hand in drei oder vier Pixel Abstand Kerning-Linien zu ziehen (verschiedene Icons für links und rechts beachten), damit Buchstaben wie M und N nicht optisch zusammenkleben. Es geht also darum, hier den Mindestabstand der Buchstaben festzulegen. Auf der Titelseite (Abb. 2) wird zusätzlich dazu noch ein "zusätzlicher Zwischenraum" (Calamus-Handbuch, I/11) eingetragen. Zu allem

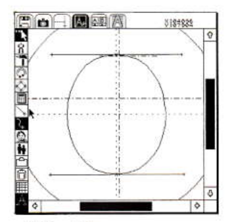

Abb.: 14

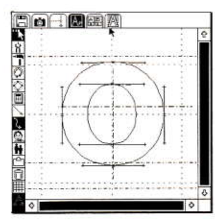

Abb.: 15

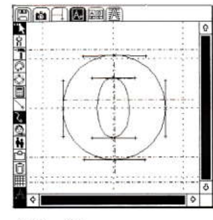

Abb.: 16


Abb.: 17


Abb.: 18


Abb.: 19


Abb.: 20


Abb.: 21

Überfluß kann man im Programm Calamus selber noch einen "Buchstabenabstand" (HandbuchIX-3/15) wählen. Woran soll man sich denn da orientieren? Nach einigem Brainstorming zusammen mit Freunden haben wir uns folgende Interpretation zurechtgelegt: Im Font-Editor auf der Kerning-Seite die individuelle Information festlegen, auf der Titelseite den "zusätzlichen" Zwischenraum und im Calamus dann die Anpassung an die verschiedenen Schriftgrößen. "Was hat das denn damit zu tun?", wird jetzt vielleicht einer fragen.

Unterschneidungen

Es geht um folgendes: Bei einer 6-Punkt-Schrift, die ja sehr klein ist, müssen zur besseren Lesbarkeit die Buchstaben etwas weiter auseinanderstehen. Bei einer 60-Punkt-Schrift müssen zur Geschlossenheit des Schriftbildes die Buchstaben enger zusammenrücken. Die Frage ist, wo man den Nullpunkt setzt (man kann im Calamuspositive und negative Buchstabenabstände eingeben). Bei den mitgelieferten Schriften Times und Swiss ist der Nullpunkt wohl bei den kleinen Schriftgraden angesetzt. Umso mehr klaffen dann z.B. 48-Punkt-Zeilen auseinander. Die muß man dann "unterschneiden" durch Eingabe negativer Werte, damit es gut aussieht.

Ich habe mich bei den Schriften, die ich erstellt habe, dafür entschieden, den Nullpunkt bei 36 Punkt anzusetzen, und deswegen auf der Titelseite des Font-Editors kleinere Werte für den "zusätzlichen Buchstabenabstand" als bei Swiss und Times eingegeben. Für eine 6-Punkt-Zeile muß dann z. B. ein Buchstabenabstand von 1.2 eingesetzt werden. Für eine 10-Punkt-Schrift ist z.B. 0.8 der richtige Wert usw.

Haben Sie Lust bekommen auf eine weitere Folge zum Font-Editor? Dann schreiben Sie uns. Sie können sich mit Problemen auch an den Autor wenden, der selbst Schriftsetzer ist und schon eine Reihe von Schriften erstellt hat:

Peter Denk
Sportzenkoppel 38
2000 Hamburg 67

Programmier Praxis

KNIGGE

Aktion sauberer Bildschirm

Andreas Hill

LANGE WURDE ER VERMISST, ENDLICH IST ER DA! WAR UNSER ATARI BIS JETZT NOCH VÖLLIG SCHUTZLOS ALLEM SCHMUTZ UND UNFLAT AUSGELIEFERT, SO GIBT ES NUN ENDLICH EINE WIRKSAME ABWEHR VON BELEIDIGUNGEN, ÜBLER NACHREDE UND HÄSSLICHEN WORTEN ALLER ART AUF UNSEREM BILDSCHIRM - DER ATARI-KNIGGE WACHT!

KNIGGE überprüft ständig den Inhalt des Tastaturpuffers und kontrolliert, ob der über die Tastatur eingegebene Text mit einem der in einer Liste stehenden Wörter/Textausschnitte übereinstimmt. Diese Liste ist hier in das Programm integriert und kann vor der Kompilierung beliebig erweitert werden. So wird etwa zuverlässig überwacht, ob der Anwender seinem Ärger über den jüngsten Bombenhagel (nach Eintippen von achtzehn Seiten Diplomarbeit mit der brandneuen Super-Wahnsinns-Textverarbeitung) mit dem Ausdruck SCH........ PROGRAMM Luft zu verschaffen sucht - nach den ersten sieben Buchstaben erscheint eine freundliche Mahnung, und nach einer angemessenen Entschuldigung ("Pardon"-Button) wird die Eingabe umgehend wieder gelöscht!

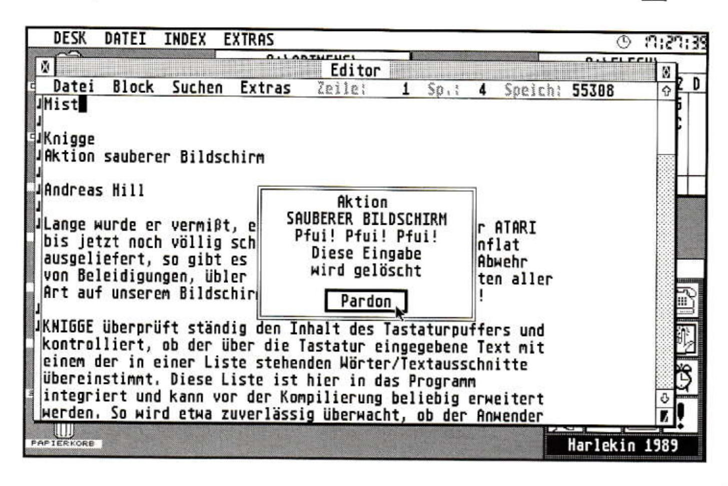
Bei KNIGGE handelt es sich um ein mit CCD Pascal plus 2.xx geschriebenes Accessory, das sich nicht in die Menüzeile einträgt. Dies läßt sich einfach durch Weglassen des *Menu_Register*-Befehls erreichen. Die Accessory-typische Endlosschleife erwartet dann nur Timer-Events, d. h. alle soundsoviel Millisekunden wird die Hauptprozedur ausgeführt, die in diesem Fall den Tastaturpuffer betrachtet und auf unflätige Eingaben hin kontrolliert. Diese Lösung ist zwar nicht so schön wie ein speicherresidentes, im Interrupt hängendes Assembler-Programm, aber dafür kann man bequem in Pascal programmieren - und Knigge läuft auch so in allen GEM-Textverarbeitungen, die Accessories unterstützen.

Wie kann ich aber nun in meinem Accessory auf Tastatureingaben warten, ohne dabei etwa die gerade laufende Textverarbeitung zu stören? Interessant ist dies sicher auch für all diejenigen, die die unzähligen (elf) Shift-Shift-Alt-Control-Tastenkombinationen leid sind. Wer gerade wieder einmal festgestellt hat, daß sowohl der Speicheraufteiler als auch der vollhydraulische Tastatur-Reset über Shift-Shift-Alternate aufgerufen werden, sucht für eigene Programme sicherlich nach Alternativen. Die Lösung wurde in einer früheren Ausgabe der "ST Computer" für C-Programmierer bereits vorgestellt: die XBIOS-Funktion *IORec*. KNIGGE ist ein Beispiel für den Umgang mit dieser in Pascal. *IORec* liefert einen Zeiger auf den Tastaturpuffer sowie Zusatzinformationen.

Der Tastaturpuffer faßt im Normalfall 64 Zeichen, auf ihm bewegen sich ein Schreib- und ein Lesezeiger. Ist eine Taste gedrückt worden, wird das Zeichen an der Stelle des Schreibzeigers im Puffer eingetragen und der Schreibzeiger weiterbewegt. Die Ausgabe auf dem Bildschirm wird über einen Vergleich von Schreib- und Lesezeiger gesteuert. Ist der Schreib-Zeiger schon weiter als der Lesezeiger, wird an der Stelle des Lesezeigers ein Zeichen aus dem Puffer geholt und der Lesezeiger weitergesetzt - solange, bis er den Schreibzeiger wieder eingeholt hat. Jedes Zeichen wird in vier Bytes abgelegt - unten der ASCII-, oben der Scan-Code. Bei Erreichen des Pufferendes wird jeweils am Anfang wieder begonnen.

Damit bei allen Vergleichen bequem mit Pascal-String-Operationen gearbeitet werden kann, unterhält KNIGGE einen eigenen Zeichenpuffer, in dem die letzten 255 Tastendrücke gespeichert werden. Das Vorhandensein neuer Zeichen wird hier über einen Vergleich von momentaner und voriger Position des Schreibzeigers geprüft, bei Ungleichheit das erste Zeichen des Puffer-Strings entfernt und dafür das neue Zeichen am Ende angehängt, Klein- in Großbuchstaben umgewandelt. Soll ein unerlaubtes Wort gelöscht werden, schreibt KNIGGE eine Sequenz von Backspace-Zeichen in den Puffer (Scan- und ASCII-Code!), setzt nach jedem den Schreibzeiger "von Hand" weiter und wartet einen Moment, damit die Textverarbeitung auch wirklich löschen kann.

Da im Tastaturpuffer alle über die Tastatur eingegebenen Zeichen stehen, wird immer nur



eine unterbrechungsfreie Zeichenkette erkannt - also etwa M, I, S, T ohne jede Betätigung einer Cursor-, Funktions- oder ähnlichen Taste. Während KNIGGE ein Wort löscht, sollten keinerlei Tastatureingaben stattfinden, da sonst nur diese neuen Zeichen eliminiert werden.

Mit KNIGGE als Grundlage kann man seine Accessories nun beispielsweise auch über beliebige Tastenkombinationen aufrufbar machen oder eine komplette Tastaturbelegung mit Makros in Pascal programmieren (Alt P gedrückt? Dies löschen, stattdessen PROGRAM in den Tastaturpuffer schreiben! usw.). Eine zusätzliche Abfrage von Control- und Shift-Taste wäre etwa über die BIOS-Funktion *KbShift* zu erreichen.

```
  1:  {$A+,S2,D-,S-,T-,P-}      { Accessory, 2 k Stack }
  2:  PROGRAM Knigge;
  3:
  4:   {  KNIGGE.ACC                                V 1.0
  5:
  6:      Demonstrationsprogramm  zur  Auswertung  und
  7:      Modifikation des Tastaturpuffers
  8:
  9:      Überwachung auf Eintippen von Schimpfwörtern
 10:      u.ä., ggf. Ausgabe einer Zurechtweisung, in-
 11:      nerhalb  von Textverarbeitungen  Löschen der
 12:      Eingabe
 13:
 14:      von Andreas Hill
 15:      (c) MAXON Computer GmbH
 16:      Letzte Modifikation: 20. April 1989
 17:
 18:      Entwickelt unter CCD Pascal plus 2.0x auf ST}
 19:
 20:   {  Linker: ACC erstellen und PASTRIX einbinden }
 21:
 22:  CONST {$I GEMCONST}          { div. GEM-Libraries   }
 23:  TYPE  {$I GEMTYPE}
 24:  {$I GEMSUBS}
 25:
 26:  CONST Zeilenlaenge = 255; { Interne Pufferlänge }
 27:        Momentchen   = 150; { ms Wartezeit        }
 28:
 29:  TYPE IORec_Type = PACKED RECORD
 30:          IBuf     : LONG_INTEGER;  { Pufferzeiger }
 31:          IBufSiz  : SHORT_INTEGER; { Pufferlänge  }
 32:          IBufHd   : SHORT_INTEGER; { Lesezeiger   }
 33:          IBufTl   : SHORT_INTEGER; { Schreibzeiger}
 34:          IBufLow  : SHORT_INTEGER;
 35:          IBufHi   : SHORT_INTEGER;
 36:          END;
 37:       Schmutzpuffer_Typ = STRING[Zeilenlaenge];
 38:
 39:  VAR Appl             : SHORT_INTEGER;
 40:      IORec_Adresse    : LONG_INTEGER;
 41:      IO_Record        : IORec_Type;
 42:      Alte_Position    : SHORT_INTEGER;
 43:      Pruefzeile       : Schmutzpuffer_Typ;
 44:      IDummy           : SHORT_INTEGER;
 45:
 46:  FUNCTION IORec (Geraet : SHORT_INTEGER)
 47:                  : LONG_INTEGER;
 48:    XBIOS (14);
 49:
 50:   { Folgende Deklarationen TRIXSUBS.PAS entnommen}
 51:
 52:  FUNCTION  LPeek (Adresse: LONG_INTEGER)
 53:                  : LONG_INTEGER;
 54:    EXTERNAL;
 55:  PROCEDURE WPoke (Adresse: LONG_INTEGER;
 56:                   Wert : SHORT_INTEGER);
 57:    EXTERNAL;
 58:  PROCEDURE LPoke (Adresse, Wert : LONG_INTEGER);
 59:    EXTERNAL;
 60:
 61:  PROCEDURE Evnt_Timer (Zeit : LONG_INTEGER);
 62:  VAR Msg : Message_Buffer;      { <Zeit> ms warten }
 63:  BEGIN
 64:     IDummy := Get_Event (E_Timer, 0,0,0, Zeit,
 65:                    False, 0,0,0,0, False, 0,0,0,0,
 66:                    Msg, IDummy, IDummy,IDummy,
 67:                    IDummy,IDummy, IDummy);
 68:  END;
 69:
 70:  PROCEDURE Peek_IORec (Adresse: LONG_INTEGER;
 71:                        VAR Rec : IORec_Type);
 72:  VAR Magic : RECORD CASE BOOLEAN OF
 73:          False: (Adr : LONG_INTEGER);
 74:          True : (Ptr : ^IORec_Type)
 75:          END;
 76:  BEGIN
 77:     Magic.Adr := Adresse;
 78:     Rec := Magic.Ptr^;
 79:  END;
 80:
 81:  PROCEDURE Pruefzeile_leeren;
 82:  VAR i : SHORT_INTEGER; { Vergleichszeile anfangs}
 83:  BEGIN                  {  mit Leerzeichen füllen}
 84:     Pruefzeile := '';
 85:     FOR i := 1 TO Zeilenlaenge DO
 86:        Pruefzeile := Concat (Pruefzeile,' ');
 87:  END;
 88:
 89:  PROCEDURE Zeichen_ausgeben (Zeichen:
 89:                              LONG_INTEGER);
 90:  BEGIN
 91:     Peek_IORec (IORec_Adresse, IO_Record);
 92:     WITH IO_Record DO BEGIN
 93:        IBufTl := IBufTl + 4; {Nächste Schreibpos.}
 94:        IF IBufTl >= IBufSiz THEN  { Pufferende ? }
 95:           IBufTl := 0;            {Dann an Anfang}
 96:        LPoke (IBuf+IBufTl,Zeichen);  { Eintragen }
 97:        WPoke (IORec_Adresse+8,IBufTl); { Zeiger->}
 98:        END;
 99:     Evnt_Timer (Momentchen); { Kontrolle abgeben }
100:  END;
101:
102:  PROCEDURE Pruefe (Schmutz : Schmutzpuffer_Typ);
103:  VAR Laenge, i : SHORT_INTEGER;
104:  BEGIN
105:     Laenge := Length (Schmutz); { Eingabe prüfen }
106:     IF Copy(Pruefzeile,Zeilenlaenge-Laenge+1,
106:             Laenge)
107:             = Schmutz THEN BEGIN
108:        IDummy := Do_Alert ('[0][        Aktion|
108:               SAUBERER BILDSCHIRM    | Pfui!
108:               Pfui! Pfui!|    Diese Eingabe|
108:               wird gelöscht][ Pardon ]',1);
109:        Evnt_Timer (Momentchen);     { Kurz warten }
110:        FOR i := 1 TO Laenge DO      { Backspaces  }
111:           Zeichen_ausgeben ($000E0008); {Scan/ASC}
112:        Write (Chr(7));               { Klingeling  }
113:        END;
114:  END;
115:
116:  PROCEDURE Pruefliste_bearbeiten;
117:  BEGIN                  {Alles in Großbuchstaben !!!}
118:     Pruefe ('MIST');
119:     Pruefe ('SCHEIP');
120:     Pruefe ('SCHEIß');
121:     Pruefe ('DOOF');
122:  END;
123:
124:  PROCEDURE IORec_Inhalt_untersuchen;
125:  VAR Zeichen      : LONG_INTEGER;
```

→

```
126:        ASCII_Code : SHORT_INTEGER;
127:   BEGIN
128:      { IO-Record lesen }
129:      Peek_IORec (IORec_Adresse, IO_Record);
130:      WITH IO_Record DO
131:         { Wurde Pufferzeiger weitergesetzt ? }
132:         IF IBufTl <> Alte_Position THEN BEGIN
133:            { Ältestes Zeichen löschen }
134:            Delete (Pruefzeile,1,1);
135:            { Neues Zeichen aus Puffer lesen }
136:            Zeichen := LPeek (IBuf+IBufTl);
137:            ASCII_Code := Int (Zeichen);
138:            { Kleine in Großbuchstaben umwandeln }
139:            IF ASCII_Code IN [97..122] THEN BEGIN
140:               ASCII_Code := ASCII_Code - 32;
141:               END;
142:            Pruefzeile := Concat (Pruefzeile,
143:                                  Chr(ASCII_Code));
144:            { Neue Zeigerposition merken }
145:            Alte_Position := IBufTl;
146:            { Alles Verbotene bemäkeln }
147:            Pruefliste_bearbeiten;
148:            END;
149:   END;
```

```
150:
151:  PROCEDURE Ereignisverwaltung;
152:  BEGIN
153:     WHILE True DO BEGIN          { Endlosschleife    }
154:        Evnt_Timer (50);          { Meldung alle 50ms}
155:        IORec_Inhalt_untersuchen;
156:        END;
157:  END;
158:
159:  BEGIN
160:     IORec_Adresse:= IORec(1); { Adresse IO-Record}
161:     Pruefzeile_leeren;          { Mit Blanks füllen}
162:
163:     Appl := Init_GEM;          {Accessory anmelden}
164:     IF Appl >= 0 THEN BEGIN    { Kein Menüeintrag }
165:        Message ('  KNIGGE ist installiert');
166:        WriteLn ('        A. Hill   4/89');
167:        WriteLn;
168:
169:        Ereignisverwaltung;      { Endlosschleife    }
170:
171:        END;
172:  END.
```

SPEICHER-VERWALTUNG

in GFA-BASIC

Manfred Müller

IRGENDWANN KAM MIR DER GEDANKE, SELBST EINEN TEXTEDITOR ZU SCHREIBEN. BEI MEINEN VORBETRACHTUNGEN KAM ICH ZU DEM SCHLUSS, DEN TEXT IN VERZEIGERTE UND GEGEN SYSTEMZUGRIFFE GESCHÜTZTE SPEICHERBLÖCKE ZU SCHREIBEN.

Normalerweise alloziert man Speicher mittels *Malloc* (GEMDOS-Funktion 72). Die TOS-Version 1.0 vom 6.2.86 gestattet aber nur ca. 280 bis 290 solcher Aufrufe (TOS-Version 1.2 vom 22.4.87 ca. 800 Aufrufe). Dies genügt für einen Texteditor auf keinen Fall, so daß ich selbst gefordert war, eine Speicherverwaltung zu implementieren.

Die Aufgaben einer Speicherverwaltung sind schnell umrissen. Auf Anforderung soll möglichst schnell ein passender Speicherblock gefunden werden, und, falls nicht vorhanden, ein größerer Block aufgespalten werden.

Bei der Rückgabe von Blöcken sollen benachbarte Speicherblöcke möglichst schnell zu einem Großen zusammengelegt werden.

Hier kam mir ein Artikel von Peter Sollich zur Hilfe, der ausführlich das Prinzip einer Speicherverwaltung behandelte. Um schnell einen der Anforderung entsprechenden Speicherblock zu finden, sortiert und verzeigert man die freien Speicher der Größe nach. Hier spricht man von der Root. Treten mehrere Blöcke mit gleicher Größe auf, so wird einer in die Root eingehängt, alle weitern enthalten nur noch Zeiger auf einen gleichgroßen. Auf der Root liegende Speicher enthalten drei Zeiger, die je vier Byte beanspruchen, plus vier Byte für den Größeneintrag. Daraus ergibt sich, daß jeder freie Block mindestens 16 Byte groß sein muß. Die verzeigerten Freispeicher nennt man Freelist. Zur Verschmelzung von benachbarten Blöcken wird eine Bitmap angelegt, in der jedes Bit 16 Byte freien Speicher repräsentiert. Die Bitmap gibt einen Überblick über den Speicher, der verwaltet wird. Jedes gesetzte Bit repräsentiert 16 Byte freien Speicher, jedes nicht gesetzte Bit 16 Byte reservierten Speicher. Bei der Prüfung auf benachbarte Blöcke muß also nur das rechte und linke Bit neben dem zurückgegebenen Block untersucht werden.

Bei der Installation der Verwaltung werden neben der Bitmap noch drei Speicherblöcke angelegt. Ein Block, der die Größe 0 repräsentiert, und ein weiterer für die Größe unendlich. Zwischen diese beiden, nennen wir sie größter und kleinster, wird nun der dritte Block eingehängt, der eigentliche freie Speicher. In der Bitmap werden die Bits für größter und kleinster gelöscht, also als reserviert markiert. Zudem wird das letzte Bit in der Bitmap auf Null gesetzt, um ein Überschreiben der Bitmap zu verhindern. Alle anderen Bits werden gesetzt.

Nun zu den Zeigern (Bild 1): Jeder Speicherblock enthält in den ersten vier Bytes seine Größe. Liegt ein freier Speicherblock auf der Root, so enthält er in Byte fünf bis acht die Startadresse des nächstgrößeren, in Byte neun bis zwölf einen Zeiger auf den nächstkleineren und schließlich in Byte dreizehn bis sechzehn die Adresse des nächsten gleichgroßen. Sollte kein gleicher Block vorhanden sein, so steht hier FALSE. Speicherblöcke, die nicht auf der Root liegen, enthalten nur den Zeiger auf den nächsten gleichen.

Zudem enthalten Blöcke, deren Größe 16 Bytes überschreiten, in den letzten vier Bytes ihre eigene Startadresse. Dieser zusätzliche Zeiger wird zur schnellen Zusammenlegung mit evtl. vorhandenen linken Nachbarn benötigt. Bei links gelegenen Nachbarn müßte in der Bitmap sonst jedes Bit, das links neben dem zurückgegebenen Block liegt, getestet werden, bis auf ein nicht gesetztes Bit getroffen wird. Aus dessen Position in der Bitmap könnte man dann die Startadresse des Nachbarn errechnen. Bei einer Speicherblockgröße von 16 kByte müßten immerhin 1000 Bits getestet werden. Soviel zum Prinzip dieser Speicherverwaltung, nun zur Implementierung in GFA-BASIC.

```
back%=FN speicher_inst(bytes%)
```

Wird hier -1 übergeben, so erhalten Sie als Rückgabe, wieviele Bytes maximal als Freispeicher installiert werden

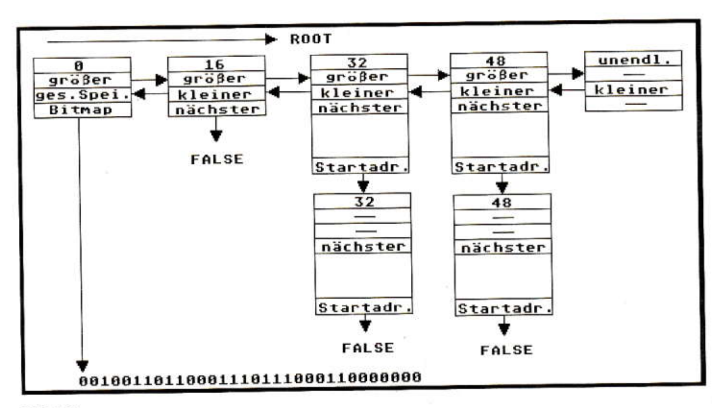


Bild 1

können. Ist der Übergabeparameter größer Null, wird die Verwaltung installiert. Bei korrekter Ausführung erhalten Sie TRUE, tritt ein Fehler auf, FALSE.

Zur Funktion selbst: In Zeile fünf wird getestet, wieviel Speicher alloziert werden kann (bei Übergabe-Parameter -1). Danach errechnet das Programm, wieviel Bytes für die Bitmap benötigt werden (allozierbarer Speicher geteilt durch 128). Das Teilen erfolgt durch Verschieben der Bits, was etwas schneller ist als dividieren. Zudem wird der Nachkommateil vernachlässigt, weil nur ganze Bytes in die Bitmap aufgenommen werden. Daraus ergibt sich, daß nur Größen, die ein Vielfaches von 128 sind, installiert werden. In Zeile sieben wird die Bitmap-Größe vom allozierbaren Speicher subtrahiert und in Zeile acht die Rückgabe errechnet. Bitmap-Größe mal 128 ergibt den maximal verwaltbaren Speicher. Davon werden je 16 Byte für größter und kleinster abgezogen. Weitere 16 Bytes werden benötigt, um das letzte Bit der Bitmap auf Null zu setzen. Wie schon erwähnt, schützt dies die Bitmap vor dem Überschreiben. In Zeile neun erfolgt der Rücksprung.

Nun zur eigentlichen Installation. In den Zeilen 11 bis 21 wird anhand des Übergabe-Parameters der zu allozierende Speicher berechnet. Danach wird mittels GEMDOS 72 Speicher vom Betriebssystem angefordert und dessen Startadresse in *malloc_adr%* abgelegt. Dies ist auch die einzige globale Variable in allen fünf Funktionen. Ab Zeile 23 werden die Größen und Zeiger eingetragen und zum Schluß die nötigen Bits in der Bitmap gesetzt. Bild 2 zeigt die fertig installierte Speicherverwaltung (Übergabeparameter 208 Bytes). Bei der Installation der Speicherverwaltung ist darauf zu achten, daß dem GEM noch genügend freier Speicher zur Verfügung steht (64 Kilobyte genügen), da sonst zum Beispiel keine Fileselectbox mehr auf dem Bildschirm erscheint. Außerdem ist eine evtl. benötigte RSC-Datei vor der Installation einzuladen.

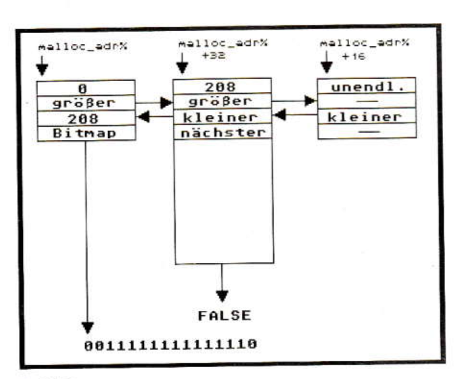


Bild 2

back%=FN speicher_res(bytes%)

Diese Function reserviert bei Aufruf die in *bytes%* angeforderte Menge Speicher und gibt dessen Startadresse zurück. Steht kein genügend großer Speicherblock zur Verfügung, wird FALSE zurückgegeben. Die beantragte Größe kann die aufrufende Anwendung in

```
     ' (c) MAXON Computer GmbH
 1:  FUNCTION speicher_frei(adr%)
 2:    LOCAL back!,bitz%,startbit%,ltest%,rtest%,
             sbyte%
 3:    LOCAL bitnr&,bitm%,radr%,ladr%,suchadr%,
             startadr%
 4:    LOCAL bytes%,startbyte2%,ltest2%,bitnr2&
 5:    SUB adr%,4
 6:    bitm%={malloc_adr%+12}
 7:    IF adr%<malloc_adr%+32 OR ODD(adr%)
 8:      back!=FALSE
 9:      RETURN back!
10:    ENDIF
11:    bytes%={adr%}
12:    ADD bytes%,4
13:    IF ROL(SHR(bytes%,4),4)<bytes%
14:      bytes%=ROL(SHR(bytes%,4),4)+16
15:    ENDIF
16:    IF adr%+bytes%>bitm%-15
17:      back!=FALSE
18:      RETURN back!
19:    ENDIF
20:    {adr%}=bytes%
21:    {malloc_adr%+8}={malloc_adr%+8}+{adr%}
22:    bitz%=SHR({adr%},4)
23:    startbit%=SHR(adr%-malloc_adr%,4)+1
24:    ltest%=startbit%-1
25:    ltest2%=startbit%-2
26:    rtest%=SHR(adr%+{adr%}-malloc_adr%,4)+1
27:    sbyte%=SHR(startbit%,3)
28:    IF ROL(SHR(startbit%,3),3)<startbit%
29:      INC sbyte%
30:    ENDIF
31:    bitnr&=ROL(sbyte%,3)-startbit%
32:    DEC sbyte%
33:    REPEAT
34:      IF bitz%>7 AND bitnr&=7
35:        BYTE{bitm%+sbyte%}=255
36:        SUB bitz%,8
37:        INC sbyte%
38:      ELSE
39:        BYTE{bitm%+sbyte%}=BSET(BYTE{bitm%+sbyte%},
                                    bitnr&)
40:        DEC bitz%
41:        DEC bitnr&
42:        IF bitnr&=-1
43:          INC sbyte%
44:          bitnr&=7
45:        ENDIF
46:      ENDIF
47:    UNTIL bitz%=0
48:    sbyte%=SHR(rtest%,3)
49:    IF ROL(SHR(rtest%,3),3)<rtest%
50:      INC sbyte%
51:    ENDIF
52:    bitnr&=ROL(sbyte%,3)-rtest%
53:    DEC sbyte%
54:    IF BTST(BYTE{bitm%+sbyte%},bitnr&)
55:      radr%=malloc_adr%+ROL(rtest%-1,4)
56:      {adr%}={adr%}+{radr%}
57:      startadr%=malloc_adr%
58:      REPEAT
59:        startadr%={startadr%+4}
60:      UNTIL {startadr%}={radr%}
61:      IF startadr%=radr%
62:        IF {radr%+12}=FALSE
63:          {{radr%+8}+4}={radr%+4}
64:          {{radr%+4}+8}={radr%+8}
65:        ELSE
66:          {{radr%+8}+4}={radr%+12}
67:          {{radr%+4}+8}={radr%+12}
68:          BMOVE radr%+4,{radr%+12}+4,8
69:        ENDIF
70:      ELSE
71:        DO
72:          EXIT IF {startadr%+12}=radr%
73:          startadr%={startadr%+12}
74:        LOOP
75:        {startadr%+12}={radr%+12}
76:      ENDIF
77:    ENDIF
78:    sbyte%=SHR(ltest%,3)
79:    IF ROL(SHR(ltest%,3),3)<ltest%
80:      INC sbyte%
81:    ENDIF
82:    bitnr&=ROL(sbyte%,3)-ltest%                  →
```

Startadresse-4 im Langwortformat auslesen.

Zum Ablauf von *speicher_res:* Zu Beginn wird die beantragte Speichergröße in *real%* zwischengespeichert. Zum Übergabeparameter werden vier aufaddiert (für Größeneintrag) und dann geprüft, ob es sich um ein Vielfaches von 16 handelt. Ist dies nicht der Fall, wird darauf erweitert. Als nächstes wird geprüft, ob der größte freie Speicher größer oder gleich *bytes%* ist. Trifft dies nicht zu, so erfolgt der Rücksprung zur Anwendung. Nun wird die Größe von *bytes%* vom Gesamtspeicher subtrahiert (steht in *malloc_adr%* +8) und danach werden die Freispeicher auf der Root durchlaufen, bis ein Block gleich oder größer *bytes%* gefunden wird. Dieses Durchlaufen der auf der Root liegenden Freispeicher wiederholt sich in einigen Functions und wird mit einer Repeat...Until Schleife realisiert. Ausgangspunkt ist dabei immer *malloc_adr%* und innerhalb der Schleife wird jedesmal der Zeiger auf den nächstgrößeren abgefragt. Ist nun ein der Anforderung passender Speicherblock gefunden, wird nach vier Fällen unterschieden:

- gleichgroß der Anforderung und keinen Zeiger auf einen gleichgroßen: Der Block wird aus der Root ausgelöst und die Startadresse in *back%* abgelegt.

- gleichgroß der Anforderung und einen Zeiger auf einen gleichgroßen: Hier wird der nächste gleich große Block ausgelöst, da nur ein Zeiger geändert werden muß. Startadresse des ausgelösten Blocks wird wieder in *back%* abgelegt.

- größer als die Anforderung und keinen Zeiger auf einen gleichen: Der Block wird aus der Root ausgelöst und die Zeiger angepaßt. Vom unteren Ende des Blockes werden der Anforderung entsprechend viele Bytes abgeschnitten und die Startadresse in *back%* abgelegt. Der verbleibende obere Teil wird wieder in die Root eingefügt.

- größer als die Anforderung und einen Zeiger auf einen gleichen: Prinzipiell wird hier genau so verfahren wie ohne nächsten.

Nun ist ein Speicherblock gefunden, ausgelöst und dessen Startadresse in *back%* abgelegt. Es muß also nur noch die Bitmap angepaßt werden. Anhand der Speicherblockgröße geteilt durch 16 wird errechnet, wieviele Bits zu löschen sind. Danach wird das Startbit errechnet, und daraus, im wievielten Byte der Bitmap mit der Löschaktion begonnen wird. Nun braucht man noch die Bit-Nummer und schon kann's losgehen. Um bei größeren Speicherblöcken eine höhere Geschwindigkeit zu erreichen, wird immer ein ganzes Byte auf Null gesetzt, falls dies möglich ist. Zum Schluß wird noch die beantragte Speicherblockgröße in die Startadresse geschrieben. Vor dem Rücksprung wird die Startadresse noch um vier erhöht, so daß die Größe in Startadresse minus vier steht und die Anwendung ab Startadresse mit dem Speicherblock tun und lassen kann, was sie will.

back%=FN speicher_frei(adr%)

Mittels speicher_frei kann schon beantragter Speicher wieder an die Verwaltung zurückgegeben werden. Übergabeparameter ist die Startadresse des als frei zu markierenden Speicherblockes. Als Rückgabe erhalten sie TRUE, tritt ein Fehler auf, FALSE.

Nun zum Ablauf: In den ersten 20 Zeilen wird auf eventuell vorhandene Fehler geprüft und die Größe wieder auf ein Vielfaches von 16 angepaßt. Danach wird errechnet, wieviele Bits in der Bitmap als frei

```
 83:     DEC sbyte%
 84:     IF BTST(BYTE{bitm%+sbyte%},bitnr&)
 85:       startbyte2%=SHR(ltest2%,3)
 86:       IF ROL(SHR(ltest2%,3),3)<ltest2%
 87:         INC startbyte2%
 88:       ENDIF
 89:       bitnr2&=ROL(startbyte2%,3)-ltest2%
 90:       DEC startbyte2%
 91:       IF BTST(BYTE{bitm%+startbyte2%},bitnr2&)
 92:         ladr%={adr%-4}
 93:       ELSE
 94:         ladr%=adr%-16
 95:       ENDIF
 96:       startadr%=malloc_adr%
 97:       REPEAT
 98:         startadr%={startadr%+4}
 99:       UNTIL {startadr%}={ladr%}
100:       IF startadr%=ladr%
101:         IF {ladr%+12}=FALSE
102:           {{ladr%+8}+4}={ladr%+4}
103:           {{ladr%+4}+8}={ladr%+8}
104:         ELSE
105:           {{ladr%+8}+4}={ladr%+12}
106:           {{ladr%+4}+8}={ladr%+12}
107:           BMOVE ladr%+4,{ladr%+12}+4,8
108:         ENDIF
109:       ELSE
110:         DO
111:           EXIT IF {startadr%+12}=ladr%
112:           startadr%={startadr%+12}
113:         LOOP
114:         {startadr%+12}={ladr%+12}
115:       ENDIF
116:       {ladr%}={ladr%}+{adr%}
117:       adr%=ladr%
118:     ENDIF
119:     IF {adr%}>16
120:       {{adr%}+adr%-4}=adr%
121:     ENDIF
122:     suchadr%=malloc_adr%
123:     REPEAT
124:       suchadr%={suchadr%+4}
125:     UNTIL {suchadr%}=>{adr%}
126:     IF {suchadr%}={adr%}
127:       {adr%+12}={suchadr%+12}
128:       {suchadr%+12}=adr%
129:     ELSE
130:       {adr%+4}=suchadr%
131:       {adr%+8}={suchadr%+8}
132:       {adr%+12}=FALSE
133:       {{adr%+8}+4}=adr%
134:       {{adr%+4}+8}=adr%
135:     ENDIF
136:     back!=TRUE
137:     RETURN back!
138:   ENDFUNC
```

Listing 1: Speicher freigeben

```
      ' (c) MAXON Computer GmbH
  1:  FUNCTION speicher_res(bytes%)
  2:    LOCAL real%,back%,adr%,such_adr%,flag!
  3:    LOCAL bitz%,startbit%,sbyte%,bitm%,bitnr&
  4:    real%=bytes%
  5:    ADD bytes%,4
  6:    IF ROL(SHR(bytes%,4),4)<bytes%
  7:      bytes%=ROL(SHR(bytes%,4),4)+16
  8:    ENDIF
  9:    IF {{malloc_adr%+24}}<bytes%
 10:      back%=FALSE
 11:      RETURN back%
 12:    ENDIF
 13:    {malloc_adr%+8}={malloc_adr%+8}-bytes%
 14:    adr%=malloc_adr%
 15:    REPEAT
 16:      adr%={adr%+4}
 17:    UNTIL {adr%}=>bytes%
 18:    IF {adr%}=bytes% AND {adr%+12}=FALSE
 19:      {{adr%+8}+4}={adr%+4}
 20:      {{adr%+4}+8}={adr%+8}
 21:      back%=adr%
 22:    ELSE IF {adr%}=bytes% AND {adr%+12}<>FALSE
 23:      back%={adr%+12}
 24:      {adr%+12}={back%+12}
```

→

markiert werden müssen und welches Bit die Startadresse der Rückgabe repräsentiert. Zugleich wird noch das Bit errechnet, das die Startadresse eines eventuell vorhandenen rechten Nachbarn repräsentiert, ebenfalls das letzte Bit des linken Nachbarn. Daraufhin wird die Rückgabe in der Bitmap als frei markiert. Nun erfolgt der Test auf einen rechten Nachbarn. Ist dieser vorhanden, so wird er aus der Freelist ausgetragen und dessen Größe zur Rückgabe addiert. Jetzt erfolgt der Test auf den linken Nachbarn. Für den Fall, daß ein linker Nachbar vorhanden ist, also das Bit links neben der Rückgabe gesetzt ist, wird zusätzlich ein Bit weiter links getestet. Ist dieses gesetzt, steht die Startadresse des linken Nachbarn in dessen letzten vier Bytes, ansonsten ist die Startadresse Rückgabeadresse minus 16. Auch dieser Block wird ausgelöst und zu dessen Größe die Rückgabegröße (die bereits um rechten Nachbarn erhöht ist) addiert. Nun braucht nur noch der ganze Block in die Freelist eingetragen werden.

```
back!=
FN speicher_shrink(adr%,ngrs%)
```

Mit dieser Funktion können Sie reservierten Speicher einschränken. Als Übergabeparameter werden die Startadresse des Speicherblocks und die neue gewünschte Größe gefordert. Als Rückgabe erhalten Sie wieder TRUE und, falls ein Fehler auftritt, FALSE. Diese Function schneidet lediglich am unteren Ende des Speicherblocks die nicht mehr benötigten Bytes ab und gibt sie mittels *FN speicher_frei* an die Verwaltung zurück.

```
back%=FN speicher_kill(modus|)
```

Mit *speicher_kill* können Sie mit Übergabeparameter 0 die Speicherverwaltung abmelden, was bedeutet, daß der in *speicher_inst* allozierte Speicher ans Betriebssystem zurückgeben wird. Als Rückgabe erhalten Sie Null, wenn kein Fehler auftritt. Bei Übergabe von eins erhalten Sie als Rückgabe die Größe des Freispeichers, bei Übergabeparameter zwei die Größe des größten zusammenhängenden Freispeichers.

```
25:    ELSE IF {adr%}>bytes% AND {adr%+12}=FALSE
26:      {{adr%+8}+4}={adr%+4}
27:      {{adr%+4}+8}={adr%+8}
28:      flag!=TRUE
29:    ELSE IF {adr%}>bytes% AND {adr%+12}<>FALSE
30:      {{adr%+8}+4}={adr%+12}
31:      {{adr%+4}+8}={adr%+12}
32:      BMOVE adr%+4,{adr%+12}+4,8
33:      flag!=TRUE
34:    ENDIF
35:    IF flag!
36:      {adr%}={adr%}-bytes%
37:      back%=adr%+{adr%}
38:      IF {adr%}>16
39:        {back%-4}=adr%
40:      ENDIF
41:      such_adr%=malloc_adr%
42:      flag!=FALSE
43:      REPEAT
44:        such_adr%={such_adr%+4}
45:        IF {such_adr%}={adr%}
46:          {adr%+12}={such_adr%+12}
47:          {such_adr%+12}=adr%
48:          flag!=TRUE
49:        ELSE IF {such_adr%}>{adr%}
50:          {adr%+8}={such_adr%+8}
51:          {adr%+4}=such_adr%
52:          {adr%+12}=FALSE
53:          {such_adr%+8}=adr%
54:          {{adr%+8}+4}=adr%
55:          flag!=TRUE
56:        ENDIF
57:      UNTIL flag!
58:    ENDIF
59:    bitz%=SHR(bytes%,4)
60:    startbit%=SHR(back%-malloc_adr%,4)+1
61:    sbyte%=SHR(startbit%,3)
62:    IF ROL(SHR(startbit%,3),3)<startbit%
63:      INC sbyte%
64:    ENDIF
65:    bitnr&=ROL(sbyte%,3)-startbit%
66:    bitm%={malloc_adr%+12}
67:    DEC sbyte%
68:    REPEAT
69:      IF bitz%>7 AND bitnr&=7
70:        BYTE{bitm%+sbyte%}=0
71:        SUB bitz%,8
72:        INC sbyte%
73:      ELSE
74:        BYTE{bitm%+sbyte%}=BCLR(BYTE{bitm%+sbyte%},
                                  bitnr&)
75:        DEC bitz%
76:        DEC bitnr&
77:        IF bitnr&=-1
78:          INC sbyte%
79:          bitnr&=7
80:        ENDIF
81:      ENDIF
82:    UNTIL bitz%=0
83:    {back%}=real%
84:    ADD back%,4
85:    RETURN back%
86:  ENDFUNC
```

Listing 2: Speicher reservieren

```
     ' (c) MAXON Computer GmbH
 1:  FUNCTION speicher_inst(bytes%)
 2:    LOCAL bitm_gr%,spei_gr%,malloc_gr%,back%
 3:    LOCAL offset%,bitm_adr%
 4:    IF bytes%=-1
 5:      malloc_gr%=MALLOC(-1)
 6:      bitm_gr%=SHR(malloc_gr%,7)
 7:      SUB malloc_gr%,bitm_gr%
 8:      back%=ROL(SHR(malloc_gr%,7),7)-48
 9:      RETURN back%
10:    ENDIF
11:    ADD bytes%,48
12:    bitm_gr%=SHR(bytes%,7)
13:    IF ROL(SHR(bytes%,7),7)<bytes%
14:      INC bitm_gr%
15:    ENDIF
16:    spei_gr%=ROL(bitm_gr%,7)
17:    malloc_gr%=spei_gr%+bitm_gr%
18:    IF MALLOC(-1)<malloc_gr% OR bitm_gr%<2
19:      back%=FALSE
20:      RETURN back%
21:    ENDIF
22:    malloc_adr%=MALLOC(malloc_gr%)
23:    {malloc_adr%}=0
24:    {malloc_adr%+16}=spei_gr%-16
25:    {malloc_adr%+12}=malloc_adr%+spei_gr%
26:    {malloc_adr%+32}=spei_gr%-48
27:    {malloc_adr%+8}=spei_gr%-48
28:    {malloc_adr%+4}=malloc_adr%+32
29:    {malloc_adr%+24}=malloc_adr%+32
30:    {malloc_adr%+36}=malloc_adr%+16
31:    {malloc_adr%+40}=malloc_adr%
32:    {malloc_adr%+44}=FALSE
33:    bitm_adr%={malloc_adr%+12}
34:    REPEAT
35:      IF bitm_gr%>3
36:        {bitm_adr%+offset%}=-1
37:        SUB bitm_gr%,4
38:        ADD offset%,4
39:      ELSE
40:        BYTE{bitm_adr%+offset%}=255
41:        DEC bitm_gr%
42:        INC offset%
43:      ENDIF
44:    UNTIL bitm_gr%=0
45:    BYTE{bitm_adr%}=63
46:    BYTE{bitm_adr%+offset%-1}=254
47:    back%=TRUE
48:    RETURN back%
49:  ENDFUNC
```

Listing 3: Speicher installieren

```
   ' (c) MAXON Computer GmbH
 1: FUNCTION speicher_kill(modus|)
 2:   LOCAL back%
 3:   SELECT modus|
 4:   CASE 0
 5:     ' Speicherverwaltung abmelden
 6:     back%=MFREE(malloc_adr%)
 7:   CASE 1
 8:     ' ges. Freispeicher
 9:     back%={malloc_adr%+8}
10:   CASE 2
11:     ' gröPter Speicherblock
12:     back%={{malloc_adr%+24}}
13:   ENDSELECT
14:   RETURN back%
15: ENDFUNC
```

Listing 4: Speicher freigeben

```
   ' (c) MAXON Computer GmbH
 1: FUNCTION speicher_shrink(adr%,ngrs%)
 2:   LOCAL agrs%,frei%,backadr%,back!,rgr%,real%
 3:   back!=FALSE
 4:   SUB adr%,4
 5:   real%=ngrs%
 6:   IF adr%<malloc_adr%+32 OR ODD(adr%)
 7:     RETURN back!
 8:   ENDIF
 9:   agrs%={adr%}
10:   rgr%=agrs%+4
11:   IF ROL(SHR(rgr%,4),4)<rgr%
12:     rgr%=ROL(SHR(rgr%,4),4)+16
13:   ENDIF
14:   IF agrs%<ngrs% OR adr%+rgr%=>{malloc_adr%+12}-
                                               15
15:     RETURN back!
16:   ENDIF
17:   ADD ngrs%,4
18:   IF ROL(SHR(ngrs%,4),4)<ngrs%
19:     ngrs%=ROL(SHR(ngrs%,4),4)+16
20:   ENDIF
21:   frei%=rgr%-ngrs%
22:   back!=TRUE
23:   IF frei%>0
24:     backadr%=adr%+ngrs%
25:     {backadr%}=frei%-4
26:     ADD backadr%,4
27:     back!=FN speicher_frei(backadr%)
28:     IF back!
29:       {adr%}=real%
30:     ENDIF
31:   ENDIF
32:   RETURN back!
33: ENDFUNC
```

Listing 5: Teil des reservierten Speichers zurückgeben

VIRSPY

Der GEMDOS-Türwächter

Gerit Gehnen

Wem ist so etwas noch nicht passiert: Man kopiert sich ein Programm auf die RAM- oder Harddisk, aber das Programm beschwert sich nach dem Starten darüber, dass noch irgendein File fehlt. Meistens liegt das an falschen Suchpfaden oder Namen, aber diese Probiererei hat jetzt ein Ende.

Mit VIRSPY hat man ein Werkzeug an der Hand, mit dem die Diskettenoperationen auf dem Drucker mitgeschrieben werden. Außerdem: Wen interessiert es nicht, ob der Lieblings-Compiler irgendwelche Zwischendateien anlegt, und wo sie erzeugt werden? Nebenbei ist VIRSPY ein Lehrstück zum Schreiben von residenten Programmen mit XBRA-Protkoll und sauberer Installation. VIRSPY fängt die GEMDOS-Aufrufe ab, die in solchen Fällen benutzt werden, nämlich *Pexec*, *Fopen*, *Fcreate* und *Sfirst*. Allerdings hatte Virspy ursprünglich eine ganz andere Aufgabe (wie schon der Name sagt), nämlich die Suche nach Link-Viren, die sich an Programme anhängen und von da aus munter weiterverbreiten. In der eigenen Routine, die zwischen Benutzer (Programm oder Desktop) und GEMDOS eingeschoben wird, werden diese Funktionen analysiert und der dazu passende Pfad- und Filename wird auf dem Drucker protokolliert. Wer keinen Drucker hat, kann die Konstante DEVICE z.B. auf den Wert 2 ändern, und sofort landen die Ausgaben auf dem Bildschirm (allerdings mit dem unschönen Effekt, daß meistens der ganze Bildschirmaufbau durcheinandergebracht wird...). Wem nach erfolgreicher Analyse das Druckergeratter zuviel wird, braucht nicht den langen Arm zum Reset-Knopf (bzw. den Dreifingergriff beim TOS 1.4) auszustrecken, sondern nur Virspy noch einmal zu starten, da dadurch ein Flag umgesetzt wird, mit dem die Druckerausgabe gesteuert wird.

Zum GEMDOS hin ist Virspy vollkommen transparent, d.h. alles, was vorne reingesteckt wird, kommt auch hinten wieder raus (wenn auch mit einer kleinen Verzögerung wegen der Druckerausgabe). Da ich ein großer Fan der XBRA-Methode bin, unterstützt Virspy natürlich auch bei der Installation das XBRA-Format und erkennt auch andere XBRA-Routinen, die nach Virspy gestartet wurden. Obwohl ich einen Mega-2 besitze, habe ich insbesondere bei der Installationsroutine auf möglichst geringen Speicherverbrauch geachtet: die Installationsroutine fliegt nämlich nach getaner Arbeit raus und hinterläßt nur einen BRA-Befehl am Anfang des Programms. Geschrieben wurde das Programm mit dem Assembler des Turbo-C, der relative Sprünge automatisch in ihre Short-Formen optimiert. Eine Anpassung an andere Assembler dürfte allerdings vollkommen unproblematisch sein. Wer sich zusätzlich ein wenig mit der Parameterübergabe ans GEMDOS auskennt, dem düfte es nicht schwerfallen, das Programm auf weitere Funktionen auszuweiten und damit noch flexibler zu machen.

```
 1:  ;  VIR_SPY  V 1.4a
 2:  ;  30. Januar 1989
 3:  ;  by  TSE
 4:  ;  Gerrit Gehnen
 5:  ;  (c) MAXON Computer GmbH
 6:  ;
 7:  ; Viren-Spion : Protokolliert alle Versuche, die
     Gemdosfunktionen
 8:  ; FOPEN, FCREATE, PEXEC,SFIRST zu benutzen auf
     dem Drucker mit.
 9:  ; Dadurch kann man leicht Zugriffe auf fremde
     Dateien ( Infektionen )
10:  ; feststellen
11:  ;
12:  ; Ausgabe:       FOPEN:  Read:    R Filename
13:  ;                        Write:  W Filename
14:  ;                        Update: U Filename
15:  ;               FCREATE:         C Filename
16:  ;               PEXEC:           E Filename (nur
                                        bei Load'n Go
                                        und Load)
17:  ;               SFIRST:          S Filename
18:  ;
19:  ; Kann durch nochmaligen Aufruf ein-und ausge-
     schaltet werden.
20:  ;
21:  ; Versionshistory (wen's interessiert):
22:  ; 1.0 (März 88) : erste Lauffähige Version
23:  ; 1.1 (April 88): Programm verkürzt und
     Deaktivierer mittels Zusatzprogramm
```

→

```
24:  ; 1.2      ''     : SFIRST mit aufgenommen
25:  ; 1.3 (3.7.88)    : Zusatzprogramm entfällt
26:  ; 1.4 (17.12.88): XBRA-Protokoll
27:  ; 1.4a (30.1.89): XBRA-Installation verbessert
                       und berichtigt
28:
29:
30:  FOPEN    equ $3d
31:  FCREAT   equ $3c
32:  EXEC     equ $4b
33:  SFIRST   equ $4e
34:  P_TERMRES equ $31
35:
36:
37:
38:
39:  DEVICE    equ 0 ; Devicenummer:0=Drucker,2=Schirm
40:
41:  CR        equ 13
42:  LF        equ 10
43:
44:            text
45:  start:    bra              install
46:  aktiv_flag:
47:            dc.w     0  ; Programmeigenes Flag für
48:                     ; Aktivierungsfunktion, kann
49:                     ; via XBRA extern manipuliert w.
50:
51:            dc.b     'XBRA'          ; XBRA-Formalismus
52:            dc.b     '4SPY'
53:  oldvect:ds.l       1
54:
55:
56:  patch:    movea.l sp,a0      ; Untersuche, ob Aufruf
    aus dem
57:            btst     #5,(a0) ; Supermodus stattfand
58:            beq              from_user
59:            addq.l   #6,a0   ; ja: Addiere Offset
60:            bra              is_it_me
61:  from_user:
62:            move.l   usp,a0
63:
64:  is_it_me:
65:            cmpi.w   #0,aktiv_flag    ; Darf ich ?
66:            bne              exit
67:            cmpi.w   #FOPEN,(a0)      ; Aber ja doch !
68:            beq              los_op   ; Bin ich
                                           gemeint ?
69:            cmpi.w   #FCREAT,(a0)
70:            beq              los_cr
71:            cmpi.w   #EXEC,(a0)
72:            beq              los_ex
73:            cmpi.w   #SFIRST,(a0)
74:            beq              los_sf
75:            bra              exit
76:
77:  los_ex:   cmpi.w   #0,2(a0)         ; Exec wurde
                                           aufgerufen:
78:            beq              con_ex   ; Ermittle, ob
                                           Load'n Go
79:            cmpi.w   #3,2(a0)         ; oder Load
                                           gemeint ist
80:            bne              exit     ; wenn nicht, wars
                                           wohl nix
81:  con_ex:   move.b   #'E',d0    ; setze Moduszeichen
82:            movea.l  4(a0),a0        ; und Filenamen
83:            bra              druckstat
84:
85:  los_cr:   move.b   #'C',d0         ; Create wurde
                                           aufgerufen:
86:            bra              loscont  ; setze Modus
                                           auf 'C' und gib
87:                                      ; Filenamen weiter
88:
89:  los_sf:   move.b   #'S',d0         ; Sfirst wurde
                                           aufgerufen:
90:                                      ; setze Modus auf
                                           'S' und gib
91:  loscont:movea.l    2(a0),a0        ; Filenamen weiter
92:            bra              druckstat
93:
94:  los_op:   move.l   d1,-(sp)        ; Open wurde
                                           aufgerufen:
95:            move.w   6(a0),d0        ; Ermittle Modus
96:            movea.l  2(a0),a0        ; Ermittle Adresse
                                           des Namens
```

→

```
97:             move.b   #'W',d1        ; Setze Modus
                                          entspr. dem
98:             cmpi.w   #1,d0          ; Aufruf
99:             beq              druckd1
100:            move.b   #'R',d1
101:            cmpi.w   #0,d0
102:            beq              druckd1
103:            move.b   #'U',d1
104:
105: druckd1:move.b      d1,d0 ; Gib das Moduszeichen in
      d1 aus
106:            move.l   (sp)+,d1
107: druckstat:
108:            bsr              druckchar  ; Alles etwas
                                              umständlich
109:            move.b   #' ',d0            ; aber dafür
                                              flexibel...
110:            bsr              druckchar  ; Space raus
111:            bsr              druckstring
112: exit:   move.l      oldvect,-(sp)      ; ..und schon
     fertig
113:            rts                     ; Rücksprung in die
                                          Orginal-
114:                                    ; GEMDOS Routine
115:
116: druckchar:
117:            movem.l  d0-d2/a0-a2,-(sp)     ; Register
                                                 retten
118:            move.w   d0,-(sp)
119:            move.w   #DEVICE,-(sp)    ; und raus damit
120:            move.w   #3,-(sp)
121:            trap     #13
122:            addq.l   #6,sp
123:            movem.l  (sp)+,d0-d2/a0-a2
124:            rts
125:
126: druckstring:
127:            move.b   (a0)+,d0         ; Gib String,
                                            der mit \0
      beendet wird
128:            cmpi.b   #0,d0            ; und an Adresse
                                            (a0) steht aus
129:            beq              loopend
130:            bsr              druckchar
131:            bra              druckstring
132: loopend:move.b      #CR,d0           ; Jetzt nur noch
                                            in neue Zeile
133:            bsr              druckchar ; gehen..
134:            move.b   #LF,d0
135:            bsr              druckchar
136:            rts
137:
138: install:pea.l       supinst          ; Supermodus zum
                                            Vektorzugriff
139:            move.w   #38,-(sp)
140:            trap     #14
141:            addq.l   #6,sp
142:            move.w   aktiv_flag,d0  ;falls ich schon
                                          installiert bin,
143:            cmpi.w   #0,d0
144:            beq              nicht_inst
145:            clr.w    -(sp)
146:            trap     #1          ;dann wars das schon.
147:
148:
149: supinst:lea.l       33*4,a1          ; Inhalt des
                                            Gemdosvektors
150:            movea.l  (a1),a0          ; holen und auf
                                            XBRA prüfen
151:            movea.l  a0,a2
152: find_next:
153:            cmpi.l   #'4SPY',-8(a2)   ; Bin ich
                                            schon da ?
154:            beq              installed ; Leider ja
155:            cmpi.l   #'XBRA',-12(a2) ; anderes XBRA-
                                            Programm da?
156:            bne              war_nix
157:            move.l   -(a2),a2         ; in der Kette
158:            bra              find_next ; Prüfe näch-
                                            stes Programm
159:
160: war_nix:move.l      (a1),(oldvect)   ; Kein Vir_spy
                                            gefunden !
161:            move.l   #patch,(a1)
162:            rts
163:
```

→

```
164:
165:  installed:
166:              eori.w  #$ffff,-14(a2)  ; Aktivflag um-
                                            setzen
167:              eori.w  #$ffff,aktiv_flag ; Eigenes Flag
                                              setzen
168:              rts
169:
170:
171:  nicht_inst:
172:              pea                meldung ; Noch nicht
                                               installiert !
173:              move    #9,-(a7)
174:              trap    #1
175:              addq.l  #6,a7
176:              clr.w   -(sp)
177:              move.l  #install-start+256,-(sp)
                  ; und resident bleiben !
178:              move.w  #P_TERMRES,-(sp)
                  ; Gag:Installationsprogramm
179:              trap    #1         ; wird gelöscht !
180:
181:              data
182:
183:  meldung:dc.b 'VIR-SPY 1.4a installiert',CR,LF
184:          dc.b '(c) MAXON Computer 1989',0
```

JOYSTICK-Abfrage in Modula

Uwe A. Ruttkamp

OKAY, ES DREHT SICH MAL WIEDER UM DEN LEIDIGEN JOYSTICK. IST ES IHNEN NOCH NICHT SO GEGANGEN: MAN MÖCHTE GERNE IN EINEM PROGRAMM DEN JOYSTICK ABFRAGEN UND STÖSST AUF EINE DER ZAHLREICHEN ASSEMBLER-PROGRAMME, DIE IN DIVERSEN ZEITUNGEN ABGEDRUCKT WORDEN SIND. IM PRINZIP IST DAS JA AUCH GANZ NETT, ABER EBEN ASSEMBLER.

Und da gibt es ein paar Gründe, weshalb man mit dieser Lösung nicht zufrieden sein könnte:

1. Die Programmiersprache, mit der man arbeitet, erlaubt keine Assembler-Einbindungen.

2. Die Assembler-Routine arbeitet nicht genauso, wie man es benötigt, und Änderungen möchte man nicht vornehmen.

3. Man möchte gerne verstehen, wie man im allgemeinen den Joystick abfragt und eine Lösung parat haben, die prinzipiell in jeder Sprache funktioniert.

Und da (trara) biete ich eine optimale Lösung an: Mein Modula-2-Modul *"JoyEvent"* beinhaltet 10 Bytes Maschinenbefehle und ist ansonsten reines Modula. Die Schnittstelle ist äußerst portabel angelegt und arbeitet nur über eine Variable. Und jetzt erkläre ich, wie es geht:

Im XBIOS gibt es eine Routine *Kbdvbase*, mit der man sich eine Tabelle einiger wichtiger Vektoren ausgeben lassen kann. Unter anderem steht hier auch der Vektor *joyvec*. An der Adresse *joyvec* liegt ein Unterprogramm, das jedesmal, wenn ein Kontakt am Joystick geöffnet oder geschlossen wird, angesprungen wird. Beim Ansprung dieses Unterprogramms übergibt XBIOS im Register A0 und auf dem Stack einen Pointer auf einen Speicherbereich, in dem die aktuellen Joystick-Daten abgelegt worden sind. Im ersten Byte steht, um welchen Port es sich handelt, im zweiten ein Wert, dessen Bits angeben, welche Bewegung stattgefunden hat.

Mich interessierte natürlich zunächst mal, was die Routine, die normalerweise vom XBIOS angesprungen wird, mit den Daten macht. Ich habe also den besagten Speicherbereich disassembliert und festgestellt: *RTS*. Die Routine macht also nichts. Ein sehr gutes Zeichen, denn das bedeutet ja schließlich, daß man, wenn man keine Register ändert und den Stack in Ruhe läßt, dort machen kann, was man will. Die einfachste und auch von mir verwendete Lösung ist also, *joyvec* auf eine Routine zu verbiegen, die so aussieht:

```
move.b  2(A0), Wert PTR
rts
```

Dieses Unterprogramm dereferenziert die Adresse, die in A0 steht (plus einem Offset von 2, da wir ja das zweite Byte des gegebenen Speicherbereichs auslesen wollen), und schreibt die sich dort befindliche Zahl an eine Adresse, die wir im Augenblick Wert *PTR* nennen wollen. Wenn man dieses kurze Stück Programm assembliert, ergibt sich folgende Speicheraufteilung:

```
13E8
0002
Wert PTR
4E75
```

Es springt einem förmlich ins Auge, daß man daraus eine Struktur machen kann von folgendem Typ:

```
RoutineRec = RECORD
  Opcode  : CARDINAL;
  Offset  : CARDINAL;
  Adresse : ADDRESS;
  Return  : CARDINAL;
END;
```

Folgendes Programmfragment ergibt sich quasi automatisch :

```
VAR
  Routine : RoutineRec;
  Wert    : BITSET;
BEGIN
  Routine.Opcode  := 13E8H;
  Routine.Offset  := 2;
  Routine.Adresse := ADR(Wert);
  Routine.Return  := 4E75H;
  ...
END;
```

Jetzt muß man nur noch an *joyvec* die Adresse von Routine. Opcode zuweisen, und schon hat man in seinem Modula-Source ein Stück Maschinensprache geschrieben, das von XBIOS bei jeder Joystick-Operation angesprochen wird und somit einen Spiegel des Wertes beschafft, der eine Aussage über die genaue Joystick-Bewegung macht. Die restlichen Anweisungen, die im Modul *JoyEvent* noch gemacht werden, sind nur eingefügt, um das Bild abzurunden und im Prinzip so verwendbar. Nur braucht man jetzt keine einzige Zeile Assembler oder gar Maschinensprache mehr, um die Joystickbewegungen zu interpretieren.
JoyEvent liefert auch noch die Angabe, ob der Feuerknopf ge-

drückt wurde. Die Antwort darauf sagt mir das VDI, da der Firebutton genauso behandelt wird wie der rechte Mausknopf. Alles andere sollte aus dem Source klar hervorgehen und in jede gewünschte Sprache portierbar sein.

```
 1:  (*---------------------------------------------*)
 2:  (*---            Module JoyEvent            ---*)
 3:  (*---            ---------------            ---*)
 4:  (*---  Modul Joystick- und Button-Abfrage   ---*)
 5:  (*---   (c) MAXON Computer GmbH             ---*)
 6:  (*--- Programmiersprache: SPC-Modula-2 V1.3 ---*)
 7:  (*--- Computersystem    : ATARI 1040 ST     ---*)
 8:  (*--- Autor             : Uwe A. Ruttkamp &
                               Clemens Fehr      ---*)
 9:  (*--- Datum             : 24.09.1988        ---*)
10:  (*---------------------------------------------*)
11:
12:  DEFINITION MODULE JoyEvent;
13:
14:  TYPE
15:   JoyEventTyp = ( Right, Left, Up, Down, None );
16:
17:   PROCEDURE InitJoyEvent;
18:    (*
19:        InitJoyEvent dient zum Initialisieren des
           JoyEvent Moduls.
20:        Von jetzt ab wird bei jeder
           Joystickbewegung intern ein
21:        Wert modifiziert. Deshalb ist es auch
           notwendig die TermJoyEvent
22:        Prozedur aufzurufen, um dieses wieder
           abzuschalten.
23:    *)
24:
25:   PROCEDURE Joystick( VAR Event : JoyEventTyp ) :
               BOOLEAN;
26:    (*
27:        Dies ist die eigentliche Kernroutine dieses
           Moduls. Durch
28:        einen Aufruf der Prozedur Joystick erfährt
           das aufrufende
29:        Programm die aktuell vom Joystick gemeldete
           Bewegung. Wenn
30:        Event gleich None ist, so befindet dich der
           Joystick im
31:        Ruhezustand. Entsprechend den anderen
           möglichen Werten wird
32:        der Joystick im Augenblick bewegt.
33:        Der Rückgabewert entspricht einem
           gedrückten Firebutton:
34:        TRUE  : Firebutton gedrückt
35:        FALSE : Firebutton nicht gedrückt
36:        Joystick liefert nur sinnvolle Werte, wenn
           zuvor ein Aufruf von
37:        InitJoyEvent stattgefunden hat.
38:    *)
39:
40:   PROCEDURE TermJoyEvent;
41:    (*
42:        Diese Prozedur muß spätstens vor Beendigung
           des laufenden
43:        Programmes aufgerufen werden. Sollte dies
           nicht geschehen, kann
44:        ein Systemabsturz im weiteren Verlauf Ihrer
           Sitzung am Atari die
45:        Folge sein.
46:        Nach einem Aufruf von TermJoyEvent liefert
           Joystick keine sinnvollen
47:        Werte mehr!
48:    *)
49:  END JoyEvent.
```

```
 1:  (*----------------------------------------------*)
 2:  (*--- Implementations Module JoyEvent        ---*)
 3:  (*--- ----------------------------------     ---*)
 4:  (*--- Modul zur Joystick- und Buttonabfrage ---*)
 5:  (*--- (c) MAXON Computer GmbH               ---*)
 6:  (*--- Programmiersprache: SPC-Modula-2 V1.3 ---*)
 7:  (*--- Computersystem    : ATARI 1040 ST     ---*)
 8:  (*--- Autor             : Uwe A. Ruttkamp &
                               Clemens Fehr      ---*)
 9:  (*--- Datum             : 24.09.1988        ---*)
10:  (*----------------------------------------------*)
11:
12:  IMPLEMENTATION MODULE JoyEvent;
13:
14:                      IMPORT XBIOS;
15:  FROM SYSTEM         IMPORT ADR, ADDRESS;
16:  FROM VDIControls IMPORT OpenVirtualWorkstation,
                             CloseVirtualWorkstation,
17:                          WorkstationInitRec,
                             WorkstationDescription;
18:  FROM VDIInputs   IMPORT MouseState, MouseCodes,
                             SampleMouseButton;
19:  FROM VDIOutputs  IMPORT Coordinate;
20:
21:  CONST
22:   MoveA0 = 13E8H;
23:   RTS    = 4E75H;
24:
25:  TYPE
26:   RoutineRec = RECORD
27:       Opcode  : CARDINAL;
28:       Offset  : CARDINAL;
29:       Adresse : ADDRESS;
30:       Return  : CARDINAL;
31:   END;
32:
33:  VAR
34:   WorkIn   : WorkstationInitRec;
35:   WorkOut  : WorkstationDescription;
36:   Handle   : INTEGER;
37:   PStatus  : MouseState;
38:   Location : Coordinate;
39:   Routine  : RoutineRec;
40:   Vector   : XBIOS.KBDVECSPtr;
41:   OldVec   : ADDRESS;
42:   Wert     : BITSET;
43:
44:   PROCEDURE InitJoyEvent;
45:   VAR
46:    i : CARDINAL;
47:   BEGIN
48:    OpenVirtualWorkstation( WorkIn, Handle,
                               WorkOut );
49:    FOR i:=0 TO 15 DO EXCL(Wert, i); END;
50:    Routine.Opcode    := MoveA0;
51:    Routine.Offset    := 2;
52:    Routine.Adresse   := ADR(Wert);
53:    Routine.Return    := RTS;
54:    (* Ab der Adresse ADR(Routine.Opcode) steht
          nun, in assemblierter
55:       Form natürlich, folgende Befehlsfolge :
56:       move.b   2(a0), Wert
57:       rts                     *)
58:    Vector            := XBIOS.Kbdvbase();
59:    OldVec            := Vector^.joyvec;
60:    Vector^.joyvec    := ADR(Routine.Opcode);
61:    (* Jetzt haben wir den Pointer, der auf die
          Routine zeigt, die bei
62:       jeder Joystickaktion angesprungen wird
          verbogen auf unsere, oben
63:       beschriebene, Routine. Diese besteht nur
          aus der Anweisung den
64:       Wert auf den A0 zeigt (plus einem Offset
          von 2) an der Stelle ab-
65:       zulegen, wo die globale Variable Wert
          steht.   *)
66:   END InitJoyEvent;
67:
68:   PROCEDURE Joystick( VAR Event : JoyEventTyp ) :
               BOOLEAN;
69:   BEGIN
70:    IF     11 IN Wert THEN Event := Right
71:     ELSIF 10 IN Wert THEN Event := Left
72:     ELSIF  9 IN Wert THEN Event := Down
73:     ELSIF  8 IN Wert THEN Event := Up
74:     ELSE  Event := None
75:    END;
76:    SampleMouseButton( Handle, PStatus, Location );
77:    RETURN (RightButton IN PStatus);
78:   END Joystick;
79:   PROCEDURE TermJoyEvent;
80:   BEGIN
81:    Vector         := XBIOS.Kbdvbase();
82:    Vector^.joyvec := OldVec;
83:    CloseVirtualWorkstation( Handle );
84:   END TermJoyEvent;
85:  END JoyEvent.
```

form_do-Routine

Eingabe mal anders

Uwe Seimet

DIE FORM_DO-ROUTINE DES AES IST BEIM ARBEITEN MIT DIALOGBOXEN EIN PRAKTISCHES HILFSMITTEL, UM DIE KOMMUNIKATION DES ATARI ST MIT DEM BENUTZER ZU ERLEICHTERN. FORM_DO ÜBERNIMMT DIE KOMPLETTE VERWALTUNG DER OBJEKTE INNERHALB EINER DIALOGBOX. WAS ABER, WENN MAN WÄHREND DES DIALOGS ZUSÄTZLICHE AKTIONEN (Z.B. EINE EIGENE TASTATURABFRAGE) DURCHFÜHREN WILL?

Oder wenn man das Aussehen von "Standardobjekten" verändern will? Hier kommt man mit der eingebauten *form_do*-Routine nicht weiter, sondern muß sich eine eigene Eingabe programmieren.

Bei der Programmierung von Dialogen nimmt uns das GEM mit seiner *form_do*-Routine einen Großteil der Arbeit ab. Ist die Dialogbox einmal per Resource Construction Set erstellt, wird nach dem Zeichnen dieses Objekts (mit Hilfe von *objc_draw*) einfach die *form_do*-Routine des AES aufgerufen, und der Dialog mit dem Benutzer läuft automatisch ab. Das Hauptprogramm erhält erst wieder die Kontrolle, wenn die Dialogbox über ein EXIT- oder TOUCHEXIT-Objekt verlassen worden ist.

Vielleicht haben Sie sich aber schon die Frage gestellt, wie es manche Programme (z.B. das DISKUS-Diskutility oder der TEMPUS-Text-Editor) realisieren, die Buttons innerhalb einer Dialogbox auch über die Tastatur zu erreichen. Dies ist nur über eine neue Eingaberoutine möglich, die die vorhandene *form_do*-Routine ersetzt und neue Funktionen ermöglicht.

Was muß eine solche Eingaberoutine leisten? Zunächst müssen Eingaben oder Änderungen, die über die Tastatur oder die Maus hervorgerufen werden, auf dem Bildschirm dargestellt werden. Wird also eine Taste gedrückt, muß das der Taste entsprechende alphanumerische Zeichen in das aktuelle Eingabefeld der Dialogbox übertragen werden. Anschließend muß der Eingabe-Cursor auf die nächste Zeichenposition gesetzt werden. Handelt es sich bei der gedrückten Taste um die Tabulator- oder eine Cursor-Taste, muß ein Wechsel der Eingabeposition oder des -feldes erfolgen. Weiterhin muß sich der Programmierer darum kümmern, daß der Cursor innerhalb von Eingabefeldern an der richtigen Stelle dargestellt wird.

Schließlich müssen auch Mausaktionen überwacht werden. Wird ein EXIT- oder TOUCHEXIT-Button angeklickt, muß der Dialog beendet werden. Handelt es sich um einen Radio-Button, muß dies insofern berücksichtigt werden, als nicht nur der Status des angeklickten Buttons geändert werden muß, sondern auch andere Radio-Buttons einen neuen Objektstatus erhalten müssen.

Sie sehen, es gibt viel zu beachten. Glücklicherweise besitzt das AES genügend Unterroutinen, um bei der Verwendung einer eigenen Dialogroutine die Überwachung der oben aufgeführten Eingaben zu erleichtern. Insbesondere spreche ich damit die Funktionen *form_keybd*, *form_button* und *objc_edit* an. In Verbindung mit diesen (leider häufig nur unzureichend dokumentierten) AES-Routinen läßt sich eine Eingaberoutine formulieren, die einige neue Möglichkeiten zur Verwaltung von Dialogen bietet und vom äußerem Erscheinungsbild her wie eine "normale" Dialogbox wirkt.

Damit die folgenden Ausführungen nicht graue Theorie

bleiben, soll anhand des Beispielprogramms DIALOG aufgezeigt werden, wie die Programmierung einer eigenen *form_do*-Routine vonstatten gehen kann. DIALOG stellt eine Dialogbox auf dem Bildschirm dar, die aus zwei Eingabefeldern, zwei Radio-Buttons sowie einem "OK"- und einem "ABBRUCH"-Button besteht. Die Buttons können wie üblich über die Maus bedient werden, und auch sonst bestehen auf den ersten Blick keine Unterschiede zu anderen Dialogboxen.

Wie sieht es auf den zweiten Blick aus? Nun, Radio-Button 1 kann nicht nur über die Maus, sondern auch über die Tastenkombination [ALTERNATE] [1] betätigt werden, Radio-Button 2 ist über [ALTERNATE][2] ansprechbar. Darüber hinaus kann der "ABBRUCH"-Button über die [Undo]-Taste erreicht werden.

Das ausführlich kommentierte Assembler-Listing zu DIALOG.PRG möchte ich nun als Grundlage nehmen, um die Programmierung einer eigenen Dialogroutine zu erläutern. Wie bereits erwähnt, bilden die AES-Routinen *form_keybd* und *form_button* das Kernstück des Programms. *form_keybd* ermöglicht es, in einer Dialogbox Tastatureingaben zu simulieren, mit *form_button* kann die Betätigung eines Mausknopfes simuliert werden. Damit man die Möglichkeit hat, während des eigentlichen Dialogs zusätzliche Aktionen vorzunehmen, geschieht die Abfrage von Tastatur und Maus nun nicht mehr über die eingebaute *form_do*-Routine des AES, die ja Manipulationen während des Dialogs nicht zuläßt, sondern über die *evnt_multi*-Routine. Allgemein können über *evnt_multi* diverse Benutzeraktionen überwacht werden. Um zu spezifizieren, worauf *evnt_multi* konkret achten soll, wird ein Parameterwort übergeben, in dem je nach zu überwachender Aktion entsprechende Bits gesetzt sind. Nach diesem AES-Aufruf überwacht GEM für uns so lange Tastatur und Maus, bis eine Aktion (Tastendruck oder Betätigung der Maustaste) seitens des Benutzers erfolgt. Erst dann wird die Programmkontrolle wieder an das Hauptprogramm zurückgegeben. Ist dies geschehen, so finden sich in einem eigens für den *evnt_multi*-Aufruf errichteten Array (in unserem Fall *ev_buff*) Angaben über die Art der Aktion. Wie bereits beim Aufruf von *evnt_multi* werden diese Daten wieder durch einzelne Bits repräsentiert.

DIALOG.PRG verzweigt nun, je nachdem, um welche Aktion es sich gehandelt hat, in eine Routine zur Behandlung von Mausaktionen oder kümmert sich um die Taste, die vom Anwender gedrückt wurde. Beschäftigen wir uns zunächst mit dem Fall, daß *evnt_multi* einen Tastendruck gemeldet hat. ASCII- und Scancode der gedrückten Taste werden uns im *ev_buff*-Array mitgeteilt. Liefert *evnt_multi* einen gültigen ASCII-Code zurück, handelt es sich um eine "normale" alphanumerische Eingabe, und das Zeichen kann direkt in die Dialogbox geschrieben werden. Ist der ASCII-Code jedoch Null, liegt also nur der Scancode der betätigten Taste vor, geht es nun darum, festzustellen, ob ein Button existiert, der durch diese Taste bedient werden kann.

Um Buttons über die Tastatur bedienen zu können, werden bei DIALOG.PRG Tastenkombinationen zusammen mit der [ALTERNATE]-Taste verwendet, da in diesem Fall von *evnt_multi* nur der Scancode der Taste, jedoch kein ASCII-Code zurückgeliefert wird. Natürlich muß man an geeigneter Stelle eine Information unterbringen, welcher Button durch welche Taste bedient werden kann. Es ist sehr praktisch, diese Zusatzinformation innnerhalb der Objektdaten der Dialogbox unterzubringen. Wie ein Objekt organisiert ist, läßt sich in Tabelle 1 sehen [1].

```
typedef struct
{
  int        ob_next;        /* -> nächstes Objekt */
  int        ob_head;        /* -> erstes Kind */
  int        ob_tail;        /* -> letztes Kind */
  unsigned int ob_type;      /* Objekttyp */
  unsigned int ob_flags;     /* Objektflags */
  unsigned int ob_state;     /* Objektstatus */
  char       *ob_spec;       /* Zeiger auf ergänzende Struktur */
  int        ob_x;           /* x-Position (rel. zum Parent-Objekt) */
  int        ob_y;           /* y-Position (rel. zum Parent-Objekt) */
  int        ob_width;       /* Breite */
  int        ob_heigt;       /* Höhe */
} OBJECT;
```

Tabelle 1: Organisation eines Objektes

Im Wort für den Objekttyp ist nur das Low-Byte belegt. Somit steht das High-Byte für eigene Anwendungen zur Verfügung. DIALOG.PRG macht sich dies zunutze, indem es bei Buttons, die über die Tastatur bedient werden können, im High-Byte des Objekttyps den Scancode der Taste erwartet, durch die der betroffene Button angesprochen werden kann.) Das Programm durchsucht nun einfach den gesamten Objektbaum auf ein Objekt, bei dem als erweiterter Objekttyp der Scancode der Taste eingetragen ist, die gedrückt wurde. Findet sich ein solches Objekt nicht, wird zum Schleifenanfang zurückgekehrt und der Dialog fortgesetzt. Ist ein Objekt vorhanden, das über die gedrückte Taste angewählt werden kann, wird die Objektnummer berechnet und über *evnt_button* ein Mausklick auf dieses Objekt simuliert. Der Effekt eines Tastendrucks in Verbindung mit der [ALTERNATE]-Taste ist somit der gleiche wie beim Anklicken eines Buttons mit der Maus.

Wurde ein alphanumerisches Zeichen (also ein Tastendruck ohne Kombination mit der [ALTERNATE]-Taste) eingegeben, wird mit den ASCII- und Scancodes für dieses Zeichen zunächt die *form_keybd*-Routine aufgerufen, die in der Lage ist, diesen Tastendruck für unsere Dialogbox aufzuarbeiten. Handelte es sich bei der gedrückten Taste um eine Cursor- oder die Tabulatortaste, liefert uns *form_keybd* die Nummer des Eingabefeldes zurück, in der wir nun den Text-Cursor positionieren müssen. Zusätzlich bekommen wir in diesem Fall eine Null als Rückgabewert, was bedeutet, daß keine Taste betätigt wurde, um deren ASCII-Darstellung wir uns kümmern müßten.

Erhalten wir keine Null zurück, sondern einen positiven Wert, so ist das Eingabefeld zwar das gleiche geblieben, aber dafür müssen wir an der aktuellen Cursor-Position ein ASCII-Zeichen eintragen. Dies geschieht mit Hilfe von *objc_edit*. Diese Funktion erlaubt es, Texteingaben in einem Formular vorzunehmen. Nach Aufruf dieser Routine mit dem ASCII-Code der gedrückten Taste wird das Zeichen in der Dialogbox dargestellt. Anschließend liefert uns *objc_edit* die neue Position des Text-Cursors zurück, der ja jetzt um eine Stelle nach rechts gerückt ist.

Falls *evnt_multi* abgebrochen wurde, weil ein Maus-Button betätigt wurde, ist zunächst zu prüfen, ob sich der Maus-Cursor überhaupt innerhalb der Dialogbox befindet. Hierzu können wir die *objc_find*-Funktion heranziehen. Die

```
typedef struct
{
  int (*ub_code) ();        /* Zeiger auf die eigene Funktion */
  long ub_parm;             /* ein optionaler Parameter */
} USERBLK;
```

Tabelle 2: Die USERBLK-Struktur

Koordinaten des Mauszeigers, die *objc_find* übergeben werden müssen, werden uns von *evnt_multi* in *ev_buff* zur Verfügung gestellt. Liefert *objc_find* einen Wert größer als Null zurück, befand sich der Mauszeiger innerhalb der Dialogbox. Andernfalls wird als Fehlersignal ein Glockenton ausgegeben, so wie es auch die "normale" *form_do*-Routine macht.

Wurde ein Objekt innerhalb der Dialogbox angeklickt, wird AES dieser Umstand mittels *form_button* mitgeteilt. Das GEM übernimmt den Rest der Arbeit, kümmert sich also z.B. darum, ob es sich beim angeklickten Objekt um einen Radio-Button handelt. In diesem Fall wird nicht nur der selektierte Button invertiert, sondern die restlichen Radio-Buttons werden deselektiert. Handelte es sich beim selektierten Objekt um einen Exit-Button, liefert *form_button* einen Wert von Null zurück. Ist dies der Fall, wird der Dialog beendet.

Wurde ein Objekt ausgewählt, das keinen Exit-Status besitzt, muß zunächst geprüft werden, ob es überhaupt selektiert werden darf und es sich nicht z.B. um einen String handelt. Stellt das angeklickte Objekt ein Eingabefeld dar, muß der Text-Cursor auf dieses Feld plaziert werden. Hier hilft die *objc_edit*-Routine weiter. Zunächst wird der Text-Cursor an der alten Stelle entfernt und anschließend an der neuen Eingabeposition dargestellt.

Nun noch zur Darstellung eigener Objekttypen, wie im Fall von DIALOG dem String, der von einer abgerundeten Box umgeben ist. Will man die Zeichenroutinen des AES umgehen und das Aussehen von Objekten weitgehend selber bestimmen, muß man mit benutzerdefinierten Objekten (G_USERDEF bzw. G_PROGDEF) arbeiten. Bei Objekten dieser Art zeigt *ob_spec* (s.o.) auf eine USER BLK-Struktur (s. Tabelle 2). Trifft das AES beim Erstellen eines Dialogs auf diesen Objekttyp, wird die Objektdarstellung nicht vom AES übernommen, sondern stattdessen eine Routine aufgerufen, deren Adresse vom Programmierer in der obigen Objektstruktur festgelegt werden kann. DIALOG.PRG nutzt nun VDI-Routinen (nur solche dürfen beim Zeichnen eigener Objekte aufgerufen werden!) dazu, Text und Rahmen des ersten Strings der Dialogbox in einem eigenen Format, also mit abgerundeten Ecken, darzustellen. Natürlich sind an dieser Stelle die verschiedensten Manipulationen denkbar.

Eine Frage ist jedoch noch offen geblieben: Wie bringe ich im High-Byte des Objekttyps eigene Informationen unter? Hier kommt es auf das verwendete Resource Construction Set an. Die Methode, den Objekttyp für eigene Daten zu nutzen, wird z.B. durch das Resource Construction Set von Kuma unterstützt. Hier hat der Anwender die Möglichkeit, für jedes Objekt einen sogenannten "erweiterten Objekttyp" (extended object type) anzugeben. Die Zahl, die man an dieser Stelle einträgt, findet sich im High-Byte des Objekttyps wieder.

Soweit meine Erläuterungen zur Programmierung einer eigenen form_do-Routine. Weitere Informationen können dem kommentierten Programm-Listing entnommen werden. Auf die GEM-Routinen, die zur Darstellung einer Dialogbox benötigt werden, bin ich nicht näher eingegangen, da diese Problematik bereits des öfteren in zurückliegenden Ausgaben des ST-Magazins angesprochen wurde. Der Einbau der vorgestellten Routine in eigene Programme gestaltet sich recht einfach, da im Grunde genommen nur der Aufruf der *form_do*-Routine des AES durch die neue Routine ersetzt werden muß.

Es dürfte keine große Schwierigkeit darstellen, die Grundidee der vorgestellten Dialogverwaltung in andere Programmiersprachen zu übertragen, da DIALOG.PRG in der Hauptsache Gebrauch von Systemroutinen macht, die ja in Hochsprachen ähnlich wie in Assembler angesprochen werden.

[1]: Jankowski, Reschke, Rabich: Atari ST Profibuch (Sybex-Verlag)



```
 1:  ******************************
 2:  *     neue form_do-Routine    *
 3:  * mit erweiterten Funktionen  *
 4:  *     1989 by Uwe Seimet      *
 5:  *  (c) MAXON Computer GmbH    *
 6:  ******************************
 7:
 8:  *für die wenigen Systemaufrufe
 9:  GEMDOS   = 1
10:  MSHRINK  = 74
11:  BIOS     = 13
12:  BCONOUT  = 3
13:
14:
15:           move.l sp,a0
16:           lea stack+400,sp ;Stackpointer initiali.
17:           move.l 4(a0),a0  ;Pointer auf Basepage
18:           move.l 12(a0),a1 ;Länge des TEXT-Segments
19:           add.l 20(a0),a1  ;Länge des DATA-Segments
20:           add.l 28(a0),a1  ;Länge des BSS-Segments
21:           lea $100(a1),a1  ;256 Bytes für Basepage
22:           move.l a1,-(sp)
23:           move.l a0,-(sp)
24:           clr -(sp)
25:           move #MSHRINK,-(sp)
26:           trap #GEMDOS     ;nicht benötigten
                                Speicher rück
27:           lea 12(sp),sp
28:           tst.l d0                  ;Fehler?
29:           bne error                 ;ja-
30:           lea intin,a5    ;Pointer auf INTIN-Array
31:           lea intout,a6   ;Pointer auf INTOUT-Array
32:           moveq #10,d0              ;appl_init
33:           move.l #$00010000,d1
34:           bsr aes
35:           bmi.s error               ;Fehler-
36:           moveq #77,d0              ;graf_handle
37:           move.l #$00050000,d1
38:           bsr aes
39:           move.l a5,a0
40:           moveq #9,d0
41:  .opn:    move #1,(a0)+
42:           dbra d0,.opn
43:           move #2,(a0)
44:           moveq #100,d0             ;v_opnvwk
45:           lea contrl+2,a3
46:           clr.l (a3)+
47:           move #11,(a3)
48:           move (a6),6(a3)           ;graf_handle
49:           bsr vdiinit
50:           move 6(a3),vdi_h  ;vdi_handle merken
51:           move #7,(a5)      ;Objektzahl eintragen
52:  fix:     moveq #114,d0             ;rsrc_obfix
53:           move.l #$01010100,d1
54:           lea objc000(pc),a0        ;Adresse der
                                         Objektdaten
55:           bsr aesobj        ;Koordinaten umrechnen
56:           subq #1,(a5)
```

→

```
57:          bpl fix  ;nächstes Objekt,wenn vorhanden
58:          clr (a5)           ;Pfeil als Mauscursor
59:          moveq #78,d0               ;graf_mouse
60:          move.l #$01010100,d1
61:          bsr aes
62:          lea objc000(pc),a4 ;Adresse der Dialogbox
63:          moveq #2,d4        ;Nummer des ersten
                                 Eingabefeldes
64:          bsr.s form_do      ;Dialog ausführen
65:          moveq #19,d0               ;appl_exit
66:          move.l #$00010000,d1
67:          bsr aes
68:  error:  clr -(sp)                  ;das war's
69:          trap #GEMDOS
70:
71:  form_do:
72:          moveq #54,d0               ;form_center
73:          move.l #$00050100,d1
74:          move.l a4,a0
75:          bsr aesobj
76:          movem.l 2(a6),d5/d6 ;form_xy und form_wh
77:          clr d2                     ;FMD_START
78:          movem.l d5/d6,2(a5)
79:          movem.l d5/d6,10(a5)
80:          bsr.s form_dial
81:          moveq #1,d0     ;bis zu Ebene 1 zeichnen
82:          move.l d0,(a5)
83:          movem.l d5/d6,4(a5)
84:          moveq #42,d0               ;objc_draw
85:          move.l #$06010100,d1
86:          move.l a4,a0
87:          bsr aesobj         ;Dialogbox darstellen
88:          bsr.s _form_do     ;zur neuen Dialogroutine
89:          moveq #3,d2        ;FMD_FINISH
90:          movem.l d5/d6,2(a5)
91:          movem.l d5/d6,10(a5)
92:  form_dial:
93:          moveq #51,d0               ;form_dial
94:          move.l #$09010100,d1
95:          move d2,(a5)
96:          bra aes
97:
98:  *neue form_do-Routine, Exit-Button wird in D7
     zurückgeliefert
99:  _form_do:
100:         moveq #1,d0     ;Cursor einschalten
101:         bsr objcedit
102:         move 2(a6),d3   ;Position des Cursors
103: dloop:  move #1,2(a5)   ;warten auf Mausklick
104:         move #1,4(a5)   ;auf linken Button warten
105:         move #1,6(a5)   ;gedrückter Mausbutton
106:         moveq #25,d0               ;evnt_multi
107:         move.l #$10070100,d1
108:         move #3,(a5)    ;MU_KEYBD |MU_BUTTON
109:         lea ev_buff,a0
110:         bsr aesobj      ;Benutzeraktion abwarten
111:         btst #1,1(a6)   ;MU_BUTTON?
112:         bne.s button    ;ja-
113:         move 10(a6),d0  ;Scan- und ASCII-Code
                              der Taste
114:         tst.b d0        ;ASCII-Zeichen?
115:         bne.s noalt     ;ja-
116:         lsr #8,d0       ;Scancode ins low Byte
117:         move.l a4,a0
118: uloop:  cmp.b 6(a0),d0  ;Button für Taste
                              gefunden?
119:         beq.s default   ;ja-
120:         tst (a0)        ;letztes Objekt erreicht
121:         beq.s noalt     ;ja-
122:         lea 24(a0),a0   ;nächstes Objekt prüfen
123:         bra uloop
124: default: sub.l a4,a0
125:         move.l a0,d7
126:         divu #24,d7     ;ergibt Objektnummer
127:         move #1,12(a6)  ;für Simulation Mausklick
128:         bra.s obj
129: noalt:  move 10(a6),2(a5) ;eingegebenes Zeichen
130:         move d4,(a5)    ;aktuelles Eingabefeld
131:         clr 4(a5)
132:         moveq #55,d0    ;form_keybd
133:         move.l #$03030100,d1
134:         move.l a4,a0
135:         bsr aesobj
136:         move 2(a6),d7   ;neues Eingabefeld
137:         tst (a6)        ;Exit-Objekt betätigt?
138:         beq exit        ;ja-
139:         tst 4(a6)       ;Wechsel des Eingabefeldes
140:         beq.s newinput   ;ja-
141:         move 10(a6),2(a5) ;eingegebenes Zeichen
142:         move d3,4(a5)   ;aktuelle Cursorposition
143:         moveq #2,d0     ;Zeichen ausgeben
144:         bsr.s objcedit
145:         move 2(a6),d3   ;neue Cursorposition
146:         bra dloop
147: button: clr (a5)
148:         move #1,2(a5)   ;bis Ebene 1 suchen
149:         move.l 2(a6),4(a5)        ;aktuelle
                                       Mausposition
150:         moveq #43,d0               ;objc_find
151:         move.l #$04010100,d1
152:         move.l a4,a0
153:         bsr.s aesobj
154:         move (a6),d7    ;Objektnummer
155:         bpl.s obj       ;Objekt wurde gefunden-
156:         move #$07,-(sp);BEL
157:         move #2,-(sp)
158:         move #BCONOUT,-(sp)
159:         trap #BIOS      ;Glockenton als
                              Fehlermeldung
160:         addq.l #6,sp
161:         bra dloop
162: obj:    moveq #56,d0    ;form_button
163:         move.l #$02020100,d1
164:         move d7,(a5)    ;angeklicktes Objekt
165:         move 12(a6),2(a5)         ;Anzahl der
                                       Mausklicks
166:         move.l a4,a0
167:         bsr.s aesobj
168:         tst (a6)        ;wurde Exit-Objekt klickt
169:         beq.s exit      ;ja-
170:         move d7,d0
171:         mulu #24,d0
172:         btst #3,9(a4,d0) ;Objekt editierbar?
173:         beq dloop       ;nein-
174:         cmp d7,d4       ;Wechsel des Eingabefeldes
175:         beq dloop       ;nein-
176: newinput:bsr.s exit     ;Cursor abschalten
177:         move d7,d4      ;neues Eingabefeld
178:         bra _form_do    ;Dialog fortführen
179: exit:   move d3,4(a5)   ;Cursorpos.
180:         moveq #3,d0
181: objcedit:move d4,(a5)   ;Nummer des Eingabefeldes
182:         move d0,6(a5)
183:         moveq #46,d0    ;objc_edit
184:         move.l #$04020100,d1
185:         move.l a4,a0
186: aesobj: move.l a0,addrin
187: aes:    lea contrl,a0
188:         move d0,(a0)
189:         movep.l d1,3(a0)
190:         move.l #aespb,d1
191:         move #$c8,d0
192:         trap #2         ;AES-Aufruf
193:         rts
194:
195: vdi:    move vdi_h,contrl+12
196: vdiinit: move d0,contrl
197:         move.l #vdipb,d1
198:         moveq #$73,d0
199:         trap #2         ;VDI-Aufruf
200:         rts
201:
202: *Benutzerdefiniertes Objekt zeichnen
203: draw:
204:         move.l 4(sp),a0
205:         move.l 10(a0),d0  ;x/y-Koordinaten
206:         sub.l #$00030000,d0
207:         move.l d0,ptsin
208:         add.l #$00060000,d0
209:         add.l 14(a0),d0
210:         move.l d0,ptsin+4
211:         lea contrl,a0
212:         move.l #$02000000,d0
213:         movep.l d0,3(a0)
214:         move #8,contrl+10
215:         moveq #11,d0    ;v_rbox
216:         bsr vdi         ;abgerundete Box zeichnen
217:         move.l 4(sp),a0
218:         move.l 26(a0),a1 ;Pointer auf Textstring
219:         lea intin,a2
220:         clr d0
221:         clr d1
```

→

```
222:  .loop:   move.b (a1)+,d0
223:           beq.s .end
224:           move d0,(a2)+
225:           addq #1,d1
226:           bra .loop
227:  .end:    move.l #$01000000,d0
228:           lea contrl,a1
229:           movep.l d0,3(a1)
230:           move d1,contrl+6
231:           move.l 10(a0),ptsin
232:           add #17,ptsin+2
233:           moveq #8,d0              ;v_gtext
234:           bra vdi
235:
236:  user:    dc.l draw
237:           dc.l spec000
238:
239:  *Daten für die Dialogbox
240:  *----------------------------------------------
241:  *verwendete Objekttypen
242:  BOX      = 20
243:  BUTTON   = 26
244:  STRING   = 24
245:  FTEXT    = 29
246:
247:
248:  objc000:.dc.w $ffff
249:          .dc.w $0001,$0007
250:          .dc.w BOX
251:          .dc.w $0000,$0010
252:          .dc.l $00021100
253:          .dc.w $0007,$0004
254:          .dc.w $002b,$000f
255:
256:          .dc.w $0002
257:          .dc.w $ffff,$ffff
258:          .dc.w STRING
259:          .dc.w $0000,$0011
260:          .dc.l user    ;" Testdialog für neue form_
                          do-Routine"
261:          .dc.w $0003,$0001
262:          .dc.w $0025,$0801
263:
264:          .dc.w $0003
265:          .dc.w $ffff,$ffff
266:          .dc.w FTEXT
267:          .dc.w $0008,$0000
268:          .dc.l spec001           ;"Eingabefeld 1:
                                      __________"
269:          .dc.w $0009,$0005
270:          .dc.w $0019,$0001
271:
272:          .dc.w $0004
273:          .dc.w $ffff,$ffff
274:          .dc.w FTEXT
275:          .dc.w $0008,$0000
276:          .dc.l spec002           ;"Eingabefeld 2:
                                      __________"
277:          .dc.w $0009,$0007
278:          .dc.w $0019,$0001
279:
280:          .dc.w $0005
281:          .dc.w $ffff,$ffff
282:          .dc.w $781a
283:          .dc.w $0011,$0001
284:          .dc.l spec003           ;"Radiobutton 1"
285:          .dc.w $0007,$000a
286:          .dc.w $000e,$0001
287:
288:          .dc.w $0006
289:          .dc.w $ffff,$ffff
290:          .dc.w $791a
291:          .dc.w $0011,$0000
292:          .dc.l spec004           ;"Radiobutton 2"
293:          .dc.w $0016,$000a
294:          .dc.w $000e,$0001
295:
296:          .dc.w $0007
297:          .dc.w $ffff,$ffff
298:          .dc.w BUTTON
299:          .dc.w $0007,$0000
300:          .dc.l spec005           ;"OK"
301:          .dc.w $000c,$000d
302:          .dc.w $0009,$0001
303:
304:          .dc.w $0000
305:          .dc.w $ffff,$ffff
306:          .dc.w $611a
307:          .dc.w $0025,$0000
308:          .dc.l spec006           ;"ABBRUCH"
309:          .dc.w $0016,$000d
310:          .dc.w $0009,$0001
311:
312:  spec000:.dc.b " Testdialog für neue form_do-
                 Routine",0
313:
314:  spec001:.dc.l txt001,plt001,val001
315:          .dc.w $0003
316:          .dc.w $0006
317:          .dc.w $0000
318:          .dc.w $1180
319:          .dc.w $0000
320:          .dc.w $ffff
321:          .dc.w $000b,$001a
322:  txt001: .dc.b "@          ",0
323:  plt001: .dc.b "Eingabefeld 1: __________",0
324:  val001: .dc.b "XXXXXXXXXX",0
325:
326:  spec002:.dc.l txt002,plt002,val002
327:          .dc.w $0003
328:          .dc.w $0006
329:          .dc.w $0000
330:          .dc.w $1180
331:          .dc.w $0000
332:          .dc.w $ffff
333:          .dc.w $000b,$001a
334:  txt002: .dc.b "@          ",0
335:  plt002: .dc.b "Eingabefeld 2: __________",0
336:  val002: .dc.b "XXXXXXXXXX",0
337:
338:  spec003:.dc.b "Radiobutton 1",0
339:
340:  spec004:.dc.b "Radiobutton 2",0
341:
342:  spec005:.dc.b "OK",0
343:
344:  spec006:.dc.b "ABBRUCH",0
345:
346:  *****************************
347:  * Konvertiert durch RSCCONV *
348:  *     Autor: Uwe Seimet     *
349:  *      (C) 1989 by CCD      *
350:  *****************************
351:
352:  *----------------------------------------------
353:
354:
355:           data
356:
357:  aespb:   dc.l contrl,global,intin,intout,addrin,
                addrout
358:
359:  vdipb:   dc.l contrl,intin,ptsin,intout,ptsout
360:
361:
362:           bss
363:
364:  *diverse AES- und VDI-Arrays
365:
366:  contrl:  ds.w 11
367:
368:  global:  ds.w 15
369:
370:  intin:   ds.w 64
371:
372:  intout:  ds.w 64
373:
374:  addrin:  ds.w 64
375:
376:  addrout: ds.w 64
377:
378:  ptsin:   ds.w 64
379:
380:  ptsout:  ds.w 64
381:
382:  ev_buff: ds.w 8   ;Puffer für evnt_multi
383:
384:  vdi_h:   ds.w 1   ;VDI-Handle
385:
386:           even
387:
388:  stack:   ds.l 100 ;eigener Stackbereich
```

Qualität, die bezahlbar ist...

Auto-Monitor-Switchbox: A.R.S. (automatic Resolution Selection). Das Programm wird automatisch in der richtigen Auflösung gestartet. Mit der Auto-Monitor-Switchbox können Sie über die Tastatur zwischen Monochrom und Farbmonitor umschalten oder einen Tastaturreset durchführen. Die mitgelieferte Software ist resetfest. Durch Einbinden der von uns mitgelieferten Routinen Umschaltmöglichkeit ohne RESET. Zusätzlicher BAS und Audio-Ausgang. Auto-Monitor Switchbox 59,90 DM, Auto-Monitor Switchbox Multisync 69,90 DM, weitere Modelle: von 29,90 DM bis 69,90 DM

Video Interface +: ermöglicht die Farbwiedergabe des Atari ST an einem Farbfernseher, Monitor oder Videorecorder mit Videoausgang. Zusätzlich ist eine Auto-Monitor-Switchbox mit einem Monitorausgang integriert. 159,- DM

Fordern Sie unseren Gesamtkatalog an.

HF-Modulator: zum Anschluß des Atari ST an jeden gewöhnlichen Farbfernseher. Der Ton wird über den Fernseher übertragen. 189,-DM, Aufpreis Monitorswitchbox 30,- DM

Diskettenlaufwerke: 3,5-Zoll- und 5,25-Zoll-Disketten-Laufwerke in vollendeter Qualität. Es werden nur die besten Materialien verwendet. Laufwerksgehäuse mit kratzfester Speziallackierung. 5,25-Zoll-Laufwerk incl. beige Frontblende, 40/80-Track-Umschalter und Software IBM-Atari, anschlußfertig 339,- DM, Chassis Atari modif. 239,- DM, 3,5-Zoll-Laufwerk incl. beige Frontblende mit NEAC FD 1037 oder TEAC FD 235 anschlußfertig 249,- DM, Chassis 179,- DM

Festplattenzubehör: wie SCSI Hostadapter, Einschaltverzögerungen, 1,2 m DMA Kabel etc.

STTAST II: ermöglicht den Anschluß einer beliebigen PC-(XT-)Tastatur am ST, umschaltbare Mehrfachbelegung der Tastaturbelegungen, freie Programmierbarkeit von Makros und Generieren von Start-Up-Files (mit AUTO Load), Tastaturreset, unterstützt auch PC Ditto und Rom Port Expander. 149,- DM Set: PC Tastatur mit Mikroschalter + ST Tast II 329,- DM

Abgesetzte Tastatur am ST: Tastaturgehäuse mit Spiralkabel, Treiberstufe, Resettaste und Joystickbuchsen eingebaut. Computertyp angeben. 109,- DM

Towergehäuse: nur Gehäuse oder mit kundenspezifischer Bestückung ab 398,- DM

RTS Tastaturkappen: ab 89,- DM

Leerkarte Speichererweiterung: komplett bestückt ohne RAM's. Auf 1 MB 84,90 DM, auf 2,5 MB 149,- DM, auf 2,5/4 MB 198,- DM

Speichererweiterung: komplett bestückt mit RAM's. Auf 1 MB 298,- DM, auf 2,5 MB 698,- DM, auf 2,5/4 MB (mit 2 MB bestückt) 749,- DM

Uhrmodul intern: die Bootsoftware befindet sich auf ROM's im Betriebssystem. Wichtig: Betriebssystem angeben. ROM TOS oder Blitter TOS. 119,- DM

Uhrmodul extern: incl. Treibersoftware. 89,- DM

Floppyswitchbox: ermöglicht den Anschluß von drei Laufwerken am ST. Ausgestattet mit speziellen Treibern für 3,5 und 5,25 Laufwerke. Computertyp angeben. 89,-DM.

Außerdem: Verbindungskabel, z. B. Scartkabel, Tastaturkabel Mega ST, Stecker, Buchsen, Romportpuffer, Romportexpander, Romportbuchsen u. v. m.



1 *Speichererweiterungen: steck- oder lötbare Speicherkarte, auch für Mega ST, jeder Speichererweiterung einzeln im Computer getestet.*

2 *Monitor-Switchboxen: Umschalten soft- und hardwaremäßig, direkt anstöpselbar oder mit Kabel, Tastaturreset, Kaltstart, A.R.S. auch für Multisync Monitore.*

3 *Diskettenlaufwerke: 3,5" oder 5,25" Diskettenlaufwerke. Spitzenmäßige Qualität, TEAC oder NEC, Netzteil VDE, GS, Thermosicherung, optional 2. Floppybuchse, A/B, 2/3 Schaltung, unterstützt PC Ditto und PC Speed.*

4 *Abgesetzte Tastaturen: ST Tast II — PC Tastatur am ST mit Super-Software oder Tastaturgehäuse mit Reset-Taste und Spiralkabel, Tastaturabdeckgehäuse*

- *PC Speed DM 498,-*
- *Supercharger 749,- DM*

Hard & Soft A. Herberg

Bahnhofstr. 289 · 4620 Castrop-Rauxel · ☏ (0 23 05) 1 57 64 · Fax 1 20 22

ICON-Heiten

Piktogramme näher betrachtet

Eine der augenfälligsten Eigenheiten der Erscheinungsform der grafischen Oberfläche des ATARI ST sind seine Piktogramme, also Bilder, die bestimmte Funktionen darstellen. Diese Bilder, die wir ab sofort neudeutsch *ICON* (sprich eiken) nennen wollen, sind deshalb so beliebt, weil sie "ohne Worte" den Sinn bildhaft darstellen. Daher sind Programme, die in sinnvoller Anzahl mit Icons ausgestattet werden, meist sehr einfach zu bedienen - nicht umsonst sind oft Bedientableaus von CAD-Systemen mit Symbolen versehen. Dafür gibt es jetzt aber fast umsonst einen Icon-Editor als Sonderdisk - mit diesem und dessen Hintergrundwissen wollen wir uns ein wenig beschäftigen.

Ein kleiner Ausblick

Leider werden Icons auf dem ST ein wenig stiefmütterlich behandelt, was sicherlich in erster Linie daran liegt, daß zunächst überhaupt kein Icon-Editor vorlag. Ein halbes Jahr später erschien dann ein Programm, daß sich Icon-Editor 'schimpfte' und mit dem tatsächlich Bildchen gezeichnet werden konnten. Allerdings war diese Software eine Zumutung, denn die Zeit, die ein Benutzer des mit Icons versehenen Programms später einsparen sollte, mußte der Programmierer mehrfach in die Gestaltung des Icons hineinstecken, da dieser Editor kaum Möglichkeiten bot und auch noch viel zu langsam war - schlichtweg "zum Abgewöhnen". Auch heute liefert ATARI nur einen wenig brauchbaren Icon-Editor zum Entwicklungspaket mit. Dieses Manko fiel mir schon vor mehr als zwei Jahren auf, so daß ich damals begann (als eine Art Steckenpferd), meinen eigenen Icon-Editor zu entwickeln. Vor kurzem habe ich diesen zunächst einmal als abgeschlossen freigegeben, wobei es mir nicht darauf ankommt, Geld damit zu verdienen (die Zeit, mit Utilities Geld verdienen zu wollen, ist eh vorbei), sondern daß er eine möglichst große Verbreitung findet. Daher haben wir beschlossen, den Icon-Editor den Lesern unserer Zeitschrift als SONDERDISK anzubieten. Begleitend dazu möchte ich das Thema Icon (sowie Sprites, Mäuse und Füllmuster - dazu später mehr) mal wieder etwas auffrischen und konkret Applikationen liefern, wie die von dem Editor gelieferten Daten in ihren Programmen genutzt werden können. Als Ausgaben stehen die Daten für das Resource-Construction-Set (1.4 und 2.0) von ATARI sowie in C, Pascal und binärem Datenformat (geeignet zur Weiterverarbeitung von beliebigen Sprachen wie auch BASIC) zur Verfügung. Also kann es losgehen, oder?

Subjekt, Prädikat, Objekt...

Nein, nein, lassen wir die weitere Vertiefung der deutschen Grammatik lieber außen vor, aber dennoch wollen wir uns mit dem Objekt und speziell mit einem Objekt in einer Dialogbox beschäftigen. Sicherlich ist dieses Thema schon oft genug beschrieben worden. Trotzdem möchte ich noch einmal kurz darauf eingehen: Eine Dialogbox besteht aus verschieden Teilen (Objekten) wie beispielsweise einer äußeren Box, ein paar (edierbaren) Texten, Tasten (Buttons) und auch Icons. Diese Objekte sind hierarchisch angeordnet. Dazu schauen Sie sich am besten Bild 1 an: Dort finden Sie eine Dialogbox, die zwei Boxen enthält. In diesen zwei Boxen findet man wiederum andere Objekte, wie hier beispielsweise in der linken Box Texte, in der mittleren Buttons (bleiben wir lieber bei dem englischen Wort für Tasten, was sich inzwischen eingebürgert hat) und in der rechten zwei Icons. Dabei kann man die Box 1 als *Vaterobjekt* (Mutterobjekt könnte man es natürlich auch nennen - es hat sich aber das aus dem englischen kommende 'father objekt' verbreitet) der Boxen 2, 3 und 4 nennen, wobei die wie-

Bild 1: Eine Dialogbox mit Icons

derum die *Kinder* (children) der Box 1 darstellen. Die Icons sind Kinder von Box 4, und Box 3 ist das Vaterobjekt der Buttons. Die Objekte haben auch noch eine gewisse Reihenfolge, so daß die Dialogbox in der richtigen Reihenfolge der Objekte gezeichnet werden kann (Genaueres findet man beispielsweise in ST-Computer 12/89 'Submenüs - Und es geht doch...", in den diversen ST-Ecken oder im ST-Computer Sonderheft 2). Zur Beschreibung eines Objektes existiert innerhalb des Speichers eine von Digital-Research vordefinierte *OBJECT-Struktur* (in Pascal RECORD), auf die das AES zugreift. Diese OBJECT-Struktur finden Sie in Listing 1, auf das wir kurz eingehen werden: Im ersten Eintrag steht der Verweis auf das folgende Objekt, so daß also ein Objekt mit dem nächsten verbunden ist und somit eine Kette gebildet wird - das letzte Objekt zeigt dann wieder auf den Vater(Start) zurück. Geschieht nun eine Verzweigung (Box 2 in Box 1), so handelt es sich um ein Kind des Objektes und den Start der "Kinderkette". Die Nummer dieses Objektes (Box 2) wird in *OB_HEAD* des Objektes *BOX 1* als Startobjekt eingetragen. Box 2 wird auf Box 3, Box 3 auf Box 4 und Box 4 wieder auf Box 1 zeigen. Da Box 4 das letzte Kind ist, wird in *OB_TAIL* des Vaterobjektes die Objektnummer von *BOX_4* eingetragen. In *OB_TYPE* finden wir die Art des Objektes, in *OB_FLAGS* diverse Merkmale wie *DEFAULT*, *EXIT* etc. und in *OB_STATE* den Status eines Objektes wie *SELECTED*, *CROSSED* etc. (siehe weiterführende Artikel). Die Ausmaße des Geltungsbereiches eines Objekts (meistens Umrandung bei *BOXTEXT* oder *BOX*) werden als x- und y-Koordinaten der linken oberen Ecke sowie der Breite und Höhe des Rechtecks in *OB_X*, *OB_Y*, *OB_WIDTH* und *OB_HEIGHT* angegeben. Der Eintrag *OB_SPEC* ist abhängig vom Objekttyp und zeigt beispielsweise bei einem Text-String auf den Text.

```
typedef struct object             /* Offset zum Anfang */
{                                 /*        \/        */
  int           ob_next;          /*  0 Nächstes Objekt */
  int           ob_head;          /*  2 erstes Kind (Kopf) */
  int           ob_tail;          /*  4 letzes Kind (Schwanz) */
  unsigned int  ob_type;          /*  6 Objekttyp: Box, Icon, Text... */
  unsigned int  ob_flags;         /*  8 Merkmale wie DEFAULT, EXIT ... */
  unsigned int  ob_state;         /* 10 Status wie SELECTED, CROSSED... */
  char          *ob_spec;         /* 12 Zeiger auf besondere Erweiterung */
  int           ob_x;             /* 16 linke obere Ecke, X-Koordinate */
  int           ob_y;             /* 18 linke obere Ecke, Y-koordinate */
  int           ob_width;         /* 20 Breite des Objekts */
  int           ob_height;        /* 22 Höhe des Objekts */
} OBJECT;
```

Listing 1: Die OBJECT-Struktur eines Objektes

```
typedef struct icon_block      /* Offset zum Anfang */
{                              /*        \/        */
  int *ib_pmask;               /*  0 Zeiger auf die Maskendaten */
  int *ib_pdata;               /*  4 Zeiger auf die Bilddaten */
  char *ib_ptext;              /*  8 Zeiger auf den Text des Icons */
  int ib_char;                 /* 12 Buchstabe des Icons, mit Farbe */
  int ib_xchar;                /* 14 X-Position des Buchstabens */
  int ib_ychar;                /* 16 Y-Position des Buchstabens */
  int ib_xicon;                /* 18 X-Position des Icons */
  int ib_yicon;                /* 20 Y-Position des Icons */
  int ib_wicon;                /* 22 Breite des Icons */
  int ib_hicon;                /* 24 Höhe des Icons */
  int ib_xtext;                /* 26 X-Position des Textes */
  int ib_ytext;                /* 28 Y-Position des Textes */
  int ib_wtext;                /* 30 Breite der Textbox */
  int ib_htext;                /* 32 Höhe der Textbox */
} ICONBLK;
```

Listing 2: Die ICONBLK-Struktur eines Icons

ICON-artig

Kommen wir aber zurück zum Ausgangspunkt, dem ICON. Das Icon ist eines von 13 definierten Objektarten, die bei der Erstellung von Dialogboxen oder Menüs verwendet werden dürfen. Natürlich lassen sich in der OBJECT-Struktur nicht alle für das Icon notwendigen Daten unterbringen. Daher zeigt auch *OB_SPEC* nicht wie bei einem Text-String auf einen Text, sondern auf eine weitere Struktur, die den Namen *ICONBLK-Struktur* trägt.

Die Typ-Deklaration der ICONBLK-Struktur in C finden Sie in Listing 2. Entgegen der normalen Vorgehensweise möchte ich nicht mit dem ersten, sondern mit dem zweiten Eintrag beginnen, der einen Zeiger auf die Bilddaten des Icons darstellt. Diese Bilddaten sind bit-weise wie in der monochromen Auflösung beim ST abgespeichert. Dies funktioniert deshalb so einfach, da das Icon nur zweifarbig (Bit 0 oder 1) sein kann, wobei trotzdem dem Vordergrund (Bilddaten) und dem Hintergrund (Maske) eine beliebige Farbe zugeordnet werden (siehe weiter unten). Die Bilddaten sind byte- und zeilenweise abgespeichert. Daraus ergibt sich eine Blocklänge von Breite/8*Höhe, wobei die Breite auf das nächste Vielfache von acht aufgerundet wird. Da ein Icon an einer beliebigen Stelle postiert werden kann, könnte es auch teilweise über einem anderen Objekt liegen. Sind nun alle Punkte, die in den Bilddaten nicht gesetzt wurden, durchsichtig oder haben sie Hintergrundfarbe? Lösen wir das Geheimnis auf, zumal es die meisten wahrscheinlich schon von der Mausmaske kennen:

Ist an einer Stelle in den Bilddaten ein Punkt nicht gesetzt, und fehlt dieser Punkt auch in der Maske, ist diese Stelle durchsichtig. Wird in der Maske dieser Punkt gesetzt, erscheint er in der Masken-/Hintergrundfarbe. Damit kann man beispielsweise einen Rand um das Icon ziehen, damit es nicht direkt in den unter ihm liegenden Untergrund übergeht. Ein gutes Beispiel finden Sie in Bild 2, in dem eine Kamera als Icon zu sehen ist, bei der einmal die Maske verwendet und auf der anderen Seite weggelassen wurde. Der Aufbau der Maske ist wie der der Bilddaten, und die Adresse finden Sie im *ICONBLK* unter *IB_PMASK*. Praktisch ist, daß der Icon-Editor Ihnen die Erstellung der Maske teilweise abnehmen kann, in dem er bei Bedarf eine Maske generiert, die eine Umrandung der Bilddaten entspricht.

Farbiger Buchstabe...

Wenn Sie sich an die Icons auf dem Desktop erinnern, wissen Sie bestimmt, daß zu einem Icon auch ein Text ("Laufwerk") und ein Buchstabe (A, B, C...) gehören. Einen Zeiger auf diesen Text finden Sie an der dritten Stelle des *ICONBLKs* (*IB_PTEXT*). Der Buchstabe ist im unteren Byte von *IB_CHAR* zu finden. Richtig! Demnach muß im oberen Byte auch

etwas versteckt sein. Teilen wir dieses Byte ein weiteres Mal in oberes und unteres Halb-Byte (Nibble), so werden wir im oberen Nibble die Vordergrund- und im unteren die Hintergrundfarbe finden, für die also jeweils vier Bit zur Verfügung stehen. Ein Icon muß also nicht schwarzweiß, sondern kann auch rot/grün sein. Ein kleiner Trick verhilft sogar zu drei Farben: An der Stelle, wo in der Maske und in den Bilddaten kein Punkt gesetzt ist, scheint der Hintergrund durch, der durch eine andersfarbige Box gebildet wird.

Relatives Icon

Als oben von den Koordinaten eines Objektes gesprochen wurde, ist noch nicht zur Sprache gekommen, daß diese Koordinaten immer relativ zum Vaterobjekt zu sehen und keine Absolutkoordinaten sind. Dies trifft natürlich auch für ein Icon zu. Allerdings fällt auf, daß es Koordinaten innerhalb der Objektstruktur (*ob_x,...*) und innerhalb der Icon-Blockstruktur (*ib_xicon,...*) gibt. Was hat es damit auf sich? Zunächst sollten Sie sich die Koordinaten innerhalb der Objektstruktur als Box vorstellen, die das Icon umgibt. Diese Box kann man sogar sichtbar machen, indem man dem Icon den Status *OUTLINED* in *ob_state* gibt. Diese Box ist relativ zum Vaterobjekt, wobei die Koordinaten in der Icon-Blockstruktur die relative Position des Icons bezüglich dieser Box angeben. Zunächst mag man den Sinn dieser zusätzlichen Möglichkeit nicht einsehen. Es wird allerdings verständlicher, wenn man weiß, daß die Objektbox, die durch die Koordinaten in der Objektstruktur gebildet wird, den Wirkungsbereich des Icons angibt. Das bedeutet, daß man einen großen Bereich um das Icon anhand dieser Box setzen kann, obwohl ein Icon relativ klein sein kann. Es füllt sich auch dann angesprochen, wenn der Benutzer ein wenig nebendran klickt. Normalerweise würde nun aber das Icon in der linken oberen Ecke der Box kleben (natürlich hätte man auch eine automatische Zentrierung einbauen können), was durch die Koordinaten in der Icon-Blockstruktur 'zurechtgerückt' werden kann. In Bild 3 erkennt man dies ganz gut, da hier die Box des Icons durch outlined hervorgehoben wurde. Im rechten Beispiel würde ein Anklicken des Icons kein Ereignis hervorrufen, da sich das Icon selbst nicht innerhalb des Wirkungsbereichs befindet. Vielleicht ist dem ein oder anderen Leser in Bild 3 aufgefallen, daß der Text in bezug auf das Bild des Icons nicht immer an der gleichen Stelle steht. Dies kann man durch Verändern der Textkoordinaten *IB_XTEXT* und *IB_YTEXT* erreichen. Auch die Größe der Textbox ist veränderbar, so daß sie viel größer als der eigentliche Text sein kann. Interessant ist, daß die Textboxkoordinaten relativ zur Objektbox sind, während die Koordinaten des Buchstabens (*IB_XCHAR* und *IB_YCHAR*) relativ zum Icon verwaltet werden. In der Praxis bedeutet dies, daß ein Verschieben des Objektes den gesamten Inhalt (Icon, Buchstabe, Bild) verursacht, während ein Ändern der Icon-Bildkoordinaten auch ein Verschieben des Buchstabens zur Folge hat. Diese Eigenschaft ist recht praktisch: Stellen Sie sich vor, Sie haben ein Icon gezeichnet, das eine Diskettenstation darstellt, und haben innerhalb des Bildes den Laufwerksbuchstaben postiert. Verschieben Sie nun das Icon innerhalb der Objektumrandung, wandert der Buchstabe automatisch mit und bleibt innerhalb des Bildes an der richtigen Stelle. Beachten Sie aber, daß dabei die Textposition nicht verändert wird. Die Höhe und Breite des Icons werden in *IB_WICON* und *IB_HICON* angegeben, wobei zu beachten ist, daß die Breite ein Vielfaches von acht betragen sollte.

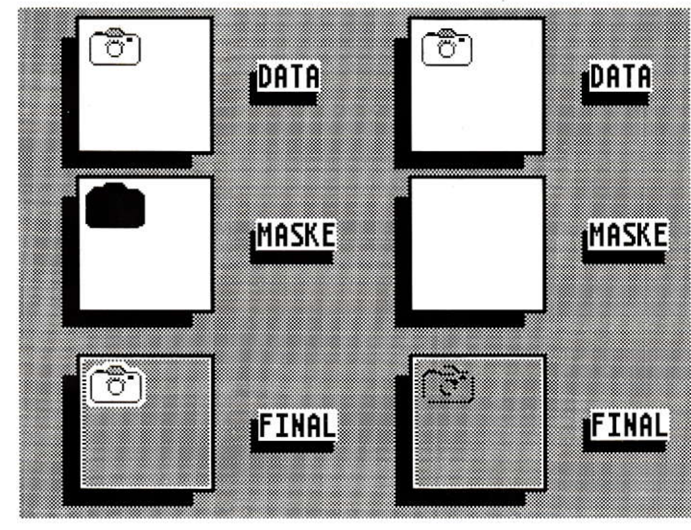


Bild 2: Die Wirkung der Maske

ICON-sinnig

An dieser Stelle können wir also zusammenfassen, daß es sich bei einem Icon um ein Objekt handelt, das ein Bild, einen Buchstaben und einen Text enthält, die alle relativ zu einer Umrandung positionierbar sind. Diese Umrandung gibt den Wirkungsbereich des Icons an, so daß das Icon im allgemeinen innerhalb des Bereich sein sollte. Wie jedem Objekt kann man auch einem Icon einen Status (*ob_status*) und Merkmale (*ob_flags*) zuordnen. Selten wird dabei die Möglichkeit der Darstellung des Wirkungsbereiches durch outlined verwendet, was aber sinnvoll sein könnte. Bis auf *EDITABLE* (ein Icon-Text ist genauso wenig edierbar wie ein normales STRING-Objekt!) können alle Einstellungen also auch beispielsweise *CROSSED* oder *CHECKED* verwendet werden.

Die ICON-tliche Anwendung

Was bringt uns ein neuer und toller Icon-Editor, wenn wir die Daten nicht auch weiterverarbeiten können? Zu diesem Zweck habe ich sehr umfangreiche Ausgaben in den Icon-Editor implementiert. Zunächst können Sie das Icon so ausgeben, daß Sie es später mit dem Resource Construction Set von ATARI (1.4 oder 2.0) wieder laden können. Dazu klicken Sie vor dem Abspeichern im Icon-Editor

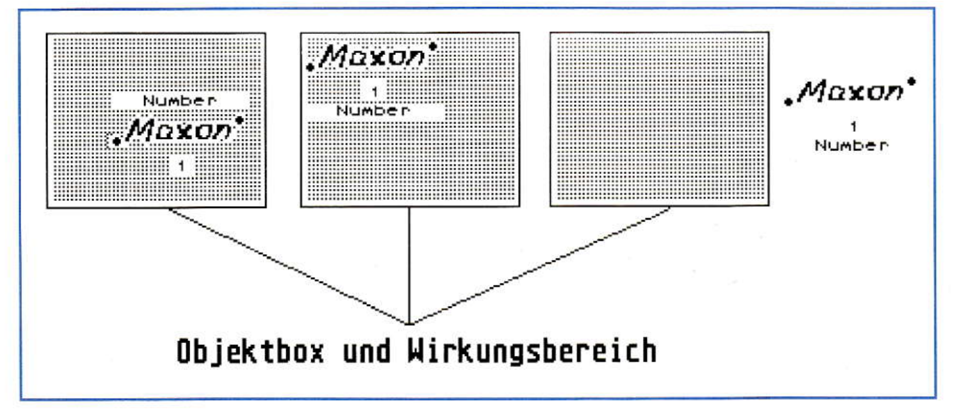


Bild 3: Der Wirkungsbereich eines Icons

unter *OPTIONEN* den Punkt *RCS-AUSGABE* an. Speichern Sie nun das erstellte Icon, werden drei Dateien erstellt: Zunächst die Icon-Editor-spezifische Datei mit dem Extender *DAT* (siehe Datenformat und Anwendung unten); danach erstellt der Editor die RCS-Dateien. Das RCS erwartet für die Maske und die Bilddaten zwei getrennte Dateien, die beide den Extender *ICN* haben sollten, da dieser beim Laden eines Icons im RCS automatisch vorgegeben wird. Deshalb wird als weiteres Unterscheidungsmerkmal in der letzten Stelle des Namens ein *D* für *DATA* oder *M* für *Mask* untergebracht. Daher sollte der Name eines Icon auch nicht länger als sieben Buchstaben sein. Das Laden eines Icons aus dem RCS geschieht nun folgendermaßen: Ziehen Sie zunächst ein vorgegebenes Icon wie ein normales Objekt in Ihre Dialogbox (oder zu erstellende Menüleiste). Danach selektieren Sie es, indem Sie mit der Maus einmal draufklicken (es sollte dadurch invertiert werden...). Wenn Sie nun in der Menüzeile des RCS nachschauen, ist ein Punkt *LOAD* selektierbar gemacht worden, der von Ihnen auch angeklickt wird. Daraufhin erscheint zweimal hintereinander eine Fileselektorbox. In der ersten Fileselektorbox wird der Name der Icon-Bilddaten- und in der zweiten der Name der Icon-Maskendatei angegeben. Wenn Sie alles richtig gemacht haben, finden Sie Ihr geladenes Icon innerhalb der Dialogbox (oder Menüzeile) wieder. Danach können Sie dieses Icon wie andere Objekte bezüglich des Status' und der Merkmale (fast) beliebig verändern (kein *EDITABLE*!). Leider ermöglichen die RCS' von ATARI, mit denen es möglich ist, ein Icon zu laden, keine großartigen Verschiebungen des Icons, des Textes und des Buchstabens - nur ein paar vordefinierte Positionen für die Positionierung des Textes und des Buchstabens sind möglich, während das Icon immer in der linken oberen Ecke 'klebt'. Schade ist es auch, daß man dem RCS keine weiteren Daten als die reinen Bild- und Maskendaten übermitteln kann. Trotzdem bietet der Icon-Editor ein Verschieben aller Teile sowie die Eingabe eines Textes und Zeichens, die zu einem Icon gehören. Diese Daten werden in einer Weise abgespeichert, die zwar nicht von einem RCS, aber sehr einfach von einem selbstgeschriebenen Programm genutzt werden. Alles, was sie dazu brauchen, finden Sie in Listing 3.

Um die Handhabung der selbsterstellten Icons möglichst einfach zu machen, habe

```
 1:  #include <obdefs.h>
 2:  #include <osbind.h>
 3:  #include "insicn.h"
 4:
 5:  main()
 6:  {
 7:     int x,y,w,h;
 8:     long baum;          /* Anfangsadresse d.Dialogbox-Objektstruktur */
 9:     long ins_icn();     /* ICON-Einfügeroutine */
10:
11:     appl_init();        /* bei AES anmelden */
12:
13:     if (!rsrc_load("insicn.rsc")) /* Demo-RSC-Datei laden */
14:     {                             /* hat nicht funktioniert */
15:        appl_exit();               /* bei AES abmelden */
16:        return(0);                 /* Programm verlassen */
17:     }
18:     rsrc_gaddr(0,BAUM,&baum);     /* Adresse der Dialogbox */
19:
20:     if (ins_icn("test.dat",baum,ICON)==-1L)  /* Einfügen des ICONs */
21:        return(-1);                         /* Bei Fehler -> Abbruch */
22:
23:     form_center(baum,&x,&y,&w,&h);              /* Box zentrieren */
24:     objc_draw(baum,0,MAX_DEPTH,x,y,x+w,y+h); /* zeichnen */
25:     form_do(baum,0);                          /* Bearbeiten */
26:
27:     clr_icn(baum,ICON);      /* Icon wieder löschen */
28:
29:     appl_exit();             /* Applikation abmelden */
30:  }
31:
32:  long ins_icn(name,tree,ob_index)    /* Einfügeroutine für Icons */
33:  char *name;                         /* Dateiname des Icons */
34:  long tree;                          /* Baumstrukturadresse */
35:  int ob_index;                       /* Objektnummer */
36:  {
37:     int fd,i=-1;
38:     ICONBLK *icon;
39:     OBJECT obj, *ob;
40:     int *data, *mask;
41:     long block_size;
42:     char strin[30], *string;
43:
44:     /* Speicherbereich in Größe eines ICONBLKs reservieren */
45:     icon= (ICONBLK*) Malloc((long)sizeof(ICONBLK));
46:     if (!icon)
47:       return(-1L);      /* Nicht genug Speicher frei -> zurück */
48:
49:     fd= Fopen(name,0);  /* Datei öffnen */
50:
51:     Fread(fd,(long)sizeof(ICONBLK),icon);  /* ICONBLK lesen */
52:     Fread(fd,(long)sizeof(OBJECT),&obj);   /* OBJECT-Strukur lesen */
53:     /* Ermitteln der Größe der Datenmenge von DATA und MASKE */
54:     block_size = (icon->ib_wicon)/8*(icon->ib_hicon);
55:     mask= (int*) Malloc(block_size); /* Maskenspeicher reservieren */
56:
57:     if (!mask)          /* kein Speicher frei */
58:     {
59:        Mfree(icon);     /* anderen Speicher wieder freigeben */
60:        return(-1L);     /* zurück */
61:     }
62:     data= (int*) Malloc(block_size); /* Speicher  für Bilddaten */
63:     if (!data)                       /* Genug Speicher da? */
64:     {
65:        Mfree(icon);   /* Nein, vorher reservierten */
66:        Mfree(mask);   /* Speicher wieder freigeben */
67:        return(-1L);
68:     }
69:     Fread(fd,block_size,mask);  /* Lesen von Maske und Bild */
70:     Fread(fd,block_size,data);
71:
72:     do             /* Icon-String einlesen */
73:     {
74:        i++;
75:        Fread(fd,1L,&strin[i]);
76:     }while (strin[i]);                  /* I = Länge des Strings */
77:     string= (char*) Malloc((long)i);  /* Speicher reservieren */
78:
79:     if (!string)        /* Speicher da? */
80:     {
81:        Mfree(icon);    /* Nein */
82:        Mfree(mask);    /* Speicher */
83:        Mfree(data);    /* freigeben */
84:        return(-1L);
85:     }
86:     strcpy(string,strin);  /* String in Speicher kopieren */      →
```

ich zwei Routinen programmiert. Die erste Routine ist in der Lage, ein Icon von Disk zu laden und dies an eine bestimmte Stelle innerhalb eines Objektbaums zu setzen. Dazu kreiert man sich die Dialogbox mit dem Standard-Icon innerhalb des RCS und gibt ihm einen Namen (in unserem Beispiel schlichtweg ICON). Beim Starten wird mit *ins_icon(dateiname, dialogboxadresse,ICON)* das zu ladende Icon an die Stelle des Standard-Icons gesetzt. Dabei werden alle Einstellungen (ICON-Position, Objekt- und Textboxgröße, Farben, Text, Buchstaben...), die Sie im Icon-Editor gemacht haben, automatisch übernommen! Außerdem wird auch der entsprechende Speicherplatz für die Daten vom Betriebssystem angefordert. Wenn Sie möchten, können Sie die Routine auch so umschreiben, daß nicht für jedes Teil (Bilddaten, Maskendaten, Icon-Blockstruktur, String) eine getrennte Speicheranforderung gemacht wird [ändern sie dann aber auch *clr_icn()*]. Wenn Sie später Ihr Programm beenden, sollten Sie mit *clr_icn(ICON)* (ICON ist wie oben der Beispielname) den reservierten Speicherbereich wieder freigeben. Wenn Sie nun beispielsweise in einem Feld die Dateinamen und die zugehörigen Objekte abspeichern, läßt sich sehr einfach in einer Schleife eine Vielzahl von Icons einlesen und in die Baumstruktur der Dialogbox einfügen.

Der Vollständigkeit halber möchte ich noch das Dateiformat angeben, was auch in *ins_icn()* abgelesen werden kann. Am Anfang steht der ICONBLK des erstellten Icons, der vollständig übernommen werden kann. Danach finden wir die Objektstruktur des Icons, wie Sie im Icon-Editor vorliegt. Hieraus verwenden wir nur die Kordinaten sowie die Flags und den Status - die Objektverknüpfungen übernehmen wir natürlich nicht. Danach folgen die Daten für die Maske und das Icon-Bild (in dieser Reihenfolge) und dann der

```
 87:
 88:     Fclose(fd);                /* Datei schließen */
 89:
 90:     ob= (OBJECT*) (tree+24*ob_index); /* Adresse des Objekts */
 91:     ob->ob_spec= (char*) icon;        /* ICONBLK eintragen */
 92:     ob->ob_width = obj.ob_width;      /* Objektumrandung */
 93:     ob->ob_height = obj.ob_height;    /* aus geladener OBJECT- */
 94:     ob->ob_state = obj.ob_state;      /* struktur übernehmen */
 95:     ob->ob_flags = obj.ob_flags;
 96:     icon->ib_pmask= mask;             /* Maskendaten eintragen */
 97:     icon->ib_pdata= data;             /* Bilddaten eintragen */
 98:     icon->ib_ptext= string;           /* String eintragen */
 99:
100:     return((long)icon);   /* Adresse auf Iconblk zurückgeben */
101: }
102:
103: clr_icn(tree,ob_index)    /* Löscht alle reservierten Speicher */
104: long tree;                /* ACHTUNG: restauriert OBJEKT nicht!! */
105: int ob_index;             /* Danach kein Aufruf des Objektes mehr! */
106: {
107:     ICONBLK *icon;        /* Zeiger auf Iconblk */
108:     OBJECT *obj;          /* Zeiger auf Objekt */
109:
110:     obj= (OBJECT*) (tree+24*ob_index);   /* Objektadresse */
111:     icon= (ICONBLK*) obj->ob_spec;       /* ICONBLK- Adresse */
112:     Mfree(icon->ib_pmask);    /* Maskenspeicher freigeben */
113:     Mfree(icon->ib_pdata);    /* Bilddaten freigeben */
114:     Mfree(icon->ib_ptext);    /* String freigeben */
115:     Mfree(icon);              /* ICONBLK freigeben */
116: }
```

Listing 3: Das Laden und Einfügen eines Icons

Text-String, der nach schöner C-Manier mit einer Null abgeschlossen ist.

Anfangs habe ich mir überlegt ein eventuell noch kompakteres Format zum Abspeichern zu wählen, allerdings ist das AES bis auf Kleinigkeiten schon so kompakt, daß es wenig zu verbessern gäbe. Außerdem hat man so die Möglichkeit, sehr einfach die erstellten Icons in eigene Programme einzubinden. Es wäre zu überlegen, ob man sich ein Programm schreibt, daß mehrere Icons zu einer Datei zusammenfügt (vielleicht zum Wiederfinden mit einem internen Namensverzeichnis) und *ins_icn()* entsprechend umschreibt. Toben Sie sich am besten ein wenig aus, und wenn Sie ein gutes Hilfsprogramm geschrieben haben und uns schicken, könnten Sie es eventuell in Zukunft auf der Icon-Editor-Sonderdisk finden.

Das nächste Mal

Mit diesem Dateiformat ist es natürlich auch sehr einfach möglich, die Icons aus BASIC heraus zu verwenden. Ein Beispiel dazu (prinzipiell ist dies die Umsetzung von Listing 3 in BASIC) finden Sie in der nächsten ST-Ecke. Dann werden wir uns mit den weiteren Möglichkeiten des Icon-Editors beschäftigen, mit dem auch Sprites, Mausbilder und Füllmuster erstellt werden können. Diese können dann als Quelltexte in PASCAL (Modula-Leute hergehört) oder C ausgegeben und damit sehr einfach in den Quelltext eingebunden werden. Man gestatte mir eine kleine Anmerkung zum Schluß: Icon mit IKONE zu übersetzen, halte ich nicht für sinnvoll; auch wenn sie noch so schön sind, göttlich sind sie bestimmt nicht...

SH

Und es geht doch!

Submenüs unter GEM

Der Nachschlag

*I*n der letzten Ausgabe der ST Computer stellten wir Ihnen eine Möglichkeit vor, mit "offiziellen" GEM-Mitteln Submenüs auf dem ATARI ST zu programmieren. Bis jetzt konnte man so etwas nur auf dem Apple Macintosh, dem AMIGA und ähnlichen Rechnern bewundern. Durch einen massiven Einsatz des berüchtigten Fehlerteufels, haben sich aber die beiden dazugehörigen Listings selbstständig gemacht, wodurch wir uns gezwungen sehen, noch einen Nachschlag zum Thema Submenüs zu geben. Hier sind sie nun, die beiden verschollenen Listings *SUBMENU.C* und *MENUDEMO.C*.

```
 1: /*************************************************/
 2: /*                 Submenu V1.15                */
 3: /*     - Einsatz von Submenus unter GEM -       */
 4: /*     - lauffähig unter allen TOS-Versionen
          und in allen Auflösungen -               */
 5: /*      (in dieser Fassung für Megamax-C)      */
 6: /*      by Uwe Hax  (c) MAXON Computer GmbH    */
 7: /*************************************************/
 8:
 9: #include <gemdefs.h>
10: #include <obdefs.h>
11: #include <osbind.h>
12: #include <portab.h>
13:
14:
15: WORD subnum; /* Rückgabewert:
                    Index des Submenüeintrags */
16:
17: /* Alle folgenden globalen Variablen werden nur
       in den folgenden Routinen benutzt und dürfen
18:    auf keinen Fall von anderen Routinen
       verändert werden!  */
19:
20: struct ext_appl_blk { /* erweiterten APPLBLK
                             definieren */
21:                     WORD (*ub_code)();
                        /* Zeiger auf
                           Zeichenroutine */
22:                     LONG ub_parm;
                        /* Index der Submenübox */
23:                     LONG regA4;
                        /* Register A4
                           (Megamax-C) */
24:                     char *text; /* Menütext */
25:                   } *block;
26:
27: WORD *menu_buffer;   /* Hintergrundspeicher für
                            Submenus */
28: WORD copy_array[8]; /* Hintergrundkoordinaten */
29: WORD fill_array[8]; /* Boxkoordinaten */
30: OBJECT *sub_menu;    /* Adresse der Submenüs */
31: FDB memory;          /* zum Kopieren */
32: FDB screen;          /* zum Kopieren */
33: WORD device;         /* VDI-Handle */
34: WORD pxyarray[4];    /* zum Zeichnen */
35: WORD mbutton,mx,my; /* Maus */
36: WORD sm_index;       /* Submenüindex */
37:
38:
39: init_submenus(handle,menu,submenu,
                  count,menu_index)
40: WORD handle;              /* VDI-Handle */
41: OBJECT *menu,*submenu; /* Adressen der beiden
                              Menüzeilen */
42: WORD count;               /* Anzahl der Elemente in
                                 der Liste */
43: WORD *menu_index;         /* Liste mit Indizes aller
                                 Menüeinträge, die Titel
44:                              f.Submenüs sein sollen
                              */
45: {
46:   WORD i;
47:   WORD draw_submenu();
48:   WORD box2;
49:   WORD *size;
50:   WORD max_size=0;
51:   WORD submenu_index;
52:   WORD menubar;
53:
54:   /* Höhe der Menüleiste ist abhängig von der
         Auflösung */
55:   menubar=(Getrez()==2) ? 19 : 11;
56:
57:   /* Speicher reservieren für alle APPLBLKs */
58:   size=(WORD *)Malloc((LONG)count*2L);
59:   block=(struct ext_appl_blk *)
60:                Malloc((LONG)(sizeof(struct
                   ext_appl_blk)*count));
61:
62:   /* Index d. G_BOX f.d. erste Submenü holen */
63:                submenu_index=submenu
                   [submenu[submenu[0].ob_tail]
                   .ob_head].ob_next;
64:
65:   for (i=0; i<count; i++)
66:   {
67:     /* Text retten */
68:     block[i].text=menu[menu_index[i]].ob_spec;
69:
70:     /* eigene Zeichenroutine installieren */
71:     menu[menu_index[i]].ob_type=G_PROGDEF;
72:
73:     /* Zeiger auf zugehörigen APPLBLK setzen */
74:     menu[menu_index[i]].ob_spec=
                          (char * )&block[i];
75:
76:     /* Adresse der neuen Zeichenroutine
           eintragen */
77:     block[i].ub_code=draw_submenu;
78:
79:     /* Parameterübergabe: Index der G_BOX des
           zugehörigen Submenüs */
80:     block[i].ub_parm=(LONG)submenu_index;
81:
82:     /* bei Eintritt in die Zeichenroutine sind
           alle Register verändert;
83:        Register A4 wird bei Megamax-C aber zur
           Adressierung der globalen
84:        Variablen benötigt */
85:     asm
86:     {
87:       move.l block(A4),A0
88:       clr.l  D0
```

→

```
 89:        move    i(A6),D0
 90:        muls    #16,D0
 91:        adda.l  D0,A0
 92:        move.l  A4,8(A0) /* A4 -> block[i].regA4 */
 93:      }
 94:
 95:      /* Index der übergeordneten G_BOX für das
            Menü suchen */
 96:      box2=menu_index[i];
 97:      while (menu[box2].ob_type!=G_BOX)
 98:        box2=menu[box2].ob_next;
 99:
100:      /* Koordinaten des Submenüs errechnen */
101:      submenu[submenu_index].ob_x=menu[box2].ob_x+
          menu[menu_index[i]].ob_width+1;
102:      submenu[submenu_index].ob_y=menubar+
          menu[menu_index[i]].ob_y;
103:
104:      /* Größe des benötigten Hintergrundspeichers
            merken */
105:      size[i]=(submenu[submenu_index].ob_width+16)

106:              /8*submenu[submenu_index].ob_height;
107:      /* nächstes Submenü */
108:      submenu_index=submenu[submenu_index].ob_next;
109:    }
110:
111:    /* Speicher für das größte Menü belegen */
112:    for (i=0; i<count; i++)
113:      if (max_size<size[i])
114:        max_size=size[i];
115:    menu_buffer=(WORD *)Malloc((LONG)max_size);
116:
117:    /* globale Variablen setzen */
118:    sub_menu=submenu;
119:    device=handle;
120:  }
121:
122:
123:  WORD draw_submenu(parmblock)
124:  PARMBLK *parmblock;
125:  {
126:    WORD index;
127:    char *sm_text;
128:    struct ext_appl_blk *appl_pointer;
129:    WORD tx,ty;
130:    WORD attrib[5];
131:    WORD state;
132:    WORD resolution;
133:    WORD correct;
134:
135:    /* Register retten */
136:    asm
137:    {
138:      movem.l D0-A6,-(A7)
139:    }
140:
141:    /* Zeiger auf den angesprochenen APPLBLK */
142:    appl_pointer=(struct ext_appl_blk *)
143:                 parmblock->pb_tree[parmblock
                     ->pb_obj].ob_spec;
144:    /* Text des Menüeintrags */
145:    sm_text=appl_pointer->text;
146:
147:    /* Register A4 für die Adressierung der
          globalen Variablen bei Megamax-C holen */
149:    asm
150:    {
151:      move.l appl_pointer(A6),A0
152:      move.l 8(A0),A4
153:    }
154:
155:    /* aktuelle Füllattribute merken */
156:    vqf_attributes(device,attrib);
157:
158:    /* Index der Submenübox */
159:    sm_index=parmblock->pb_parm;
160:
161:    /* kein Submenü angewählt */
162:    subnum=0;
163:
164:    /* Korrekturfaktor für die Ausgabe von Texten
          (abhängig von der Auflösung) */
165:    resolution=Getrez();
166:    correct=(resolution==2) ? 3 : 2;
167:
168:    /* Auf keinen Fall Clipping durchführen;
          Clip-Koordinaten sind 0!!! */
169:
170:    /* Ist der Eintrag NORMAL? */
171:    if (!parmblock->pb_currstate)     /* ja */
172:    {
173:      /* war der Eintrag vorher SELECTED? */
174:      if (parmblock->pb_prevstate)        /* ja */
175:      {
176:        /* Maus im Submenü? */
177:        vq_mouse(device,&mbutton,&mx,&my);
178:
179:        /* wenn ja, Submenükontrolle */
180:        if ((mx>=sub_menu[sm_index].ob_x-1) &&
181:            (mx<=sub_menu[sm_index].ob_x+
              sub_menu[sm_index].ob_width) &&
182:            (my>=sub_menu[sm_index].ob_y) &&
183:            (my<=sub_menu[sm_index].ob_y+
              sub_menu[sm_index].ob_height
              +1))
184:          do_submenu();
185:
186:        /* Hintergrund restaurieren */
187:        redraw_bg();
188:
189:        /* Menüeintrag neu zeichnen */
190:        pxyarray[0]=parmblock->pb_x;
191:        pxyarray[1]=parmblock->pb_y;
192:        pxyarray[2]=parmblock->pb_x+parmblock
                    ->pb_w-1;
193:        pxyarray[3]=parmblock->pb_y+parmblock
                    ->pb_h-1;
194:        switch_entry(FALSE);
195:      }
196:      else
197:      {
198:        /* Text ausgeben */
199:        v_gtext(device,parmblock->pb_x,parmblock
                ->pb_y+parmblock->pb_h-correct,
200:                sm_text);
201:      }
202:      state=NORMAL;
203:    }
204:    else      /* SELECTED */
205:    {
206:      /* Menüeintrag selektieren */
207:      pxyarray[0]=parmblock->pb_x;
208:      pxyarray[1]=parmblock->pb_y;
209:      pxyarray[2]=parmblock->pb_x+parmblock
                  ->pb_w-1;
210:      pxyarray[3]=parmblock->pb_y+parmblock
                  ->pb_h-1;
211:      switch_entry(FALSE);
212:
213:      /* Submenü zeichnen */
214:      /* ================ */
215:      /* Hintergrund retten */
216:      screen.fd_addr=0L;
217:      memory.fd_addr=(LONG)menu_buffer;
218:      memory.fd_wdwidth=sub_menu[sm_index]
          .ob_width/16+1;
219:      memory.fd_stand=0;
220:      memory.fd_nplanes=resolution ?
          2/resolution : 4;
221:      copy_array[0]=sub_menu[sm_index].ob_x;
222:      copy_array[1]=sub_menu[sm_index].ob_y-1;
223:      copy_array[2]=sub_menu[sm_index].ob_x+
          sub_menu[sm_index].ob_width+1;
224:      copy_array[3]=sub_menu[sm_index].ob_y+
          sub_menu[sm_index].ob_height;
225:      copy_array[4]=0;
226:      copy_array[5]=0;
227:      copy_array[6]=sub_menu[sm_index].ob_width+1;
228:      copy_array[7]=sub_menu[sm_index].ob_height+1;
229:      vro_cpyfm(device,3,copy_array,&screen,
          &memory);
230:
231:      /* Box zeichnen */
232:      fill_array[0]=copy_array[0];
233:      fill_array[1]=copy_array[1];
234:      fill_array[2]=copy_array[2];
235:      fill_array[3]=copy_array[1];
236:      fill_array[4]=copy_array[2];
237:      fill_array[5]=copy_array[3];
238:      fill_array[6]=copy_array[0];
239:      fill_array[7]=copy_array[3];
```

→

```
240:      vsf_interior(device,0);
241:      vswr_mode(device,1);
242:      v_fillarea(device,4,fill_array);
243:
244:      /* Texte ausgeben */
245:      index=sub_menu[sm_index].ob_head;
246:      while (sub_menu[index].ob_type==G_STRING)
247:      {
248:        if (sub_menu[index].ob_state & DISABLED)
249:          vst_effects(device,2);
250:        v_gtext(device,tx=sub_menu[sm_index].ob_x+
          sub_menu[index].ob_x+1,
251:                        ty=sub_menu[sm_index].ob_y+
                              sub_menu[index].ob_y+
252:                        sub_menu[index].ob_height-
                        correct,sub_menu[index]
                        .ob_spec);
253:        if (sub_menu[index].ob_state & CHECKED)
254:          v_gtext(device,tx+2,ty,"\10");
255:        vst_effects(device,0);
256:        index=sub_menu[index].ob_next;
257:      }
258:      state=SELECTED;
259:    }
260:
261:    /* alte Füllattribute wieder herstellen */
262:    vsf_interior(device,attrib[0]);
263:    vsf_color(device,attrib[1]);
264:    vsf_style(device,attrib[2]);
265:    vswr_mode(device,attrib[3]);
266:    vsf_perimeter(device,attrib[4]);
267:
268:    /* Register zurückholen */
269:    asm
270:    {
271:      movem.l (A7)+,D0-A6
272:    }
273:
274:    /* Zustand des angewählten Menüeintrags
          zurückgeben */
275:    return(state);
276:  }
277:
278:
279:  redraw_bg()
280:  {
281:    v_hide_c(device);
282:
283:    copy_array[0]=0;
284:    copy_array[1]=0;
285:    copy_array[2]=sub_menu[sm_index].ob_width+1;
286:    copy_array[3]=sub_menu[sm_index].ob_height+1;
287:    copy_array[4]=sub_menu[sm_index].ob_x;
288:    copy_array[5]=sub_menu[sm_index].ob_y-1;
289:    copy_array[6]=sub_menu[sm_index].ob_x+
        sub_menu[sm_index].ob_width+1;
290:    copy_array[7]=sub_menu[sm_index].ob_y+
        sub_menu[sm_index].ob_height;
291:
292:    /* alle and. Parameter sind bereits gesetzt */
293:    vro_cpyfm(device,3,copy_array,&memory,&screen);
294:
295:    v_show_c(device,1);
296:  }
297:
298:
299:  switch_entry(disabled)
300:  WORD disabled;
301:  {
302:    /* je nach übergebenem Parameter Menüeintrag
          entweder normal oder selektiert zeichnen */
304:    if (!disabled)
305:    {
306:      vsf_interior(device,1);
307:      vswr_mode(device,3);
308:      vr_recfl(device,pxyarray);
309:    }
310:  }
311:
312:  do_submenu()
313:  {
314:    WORD index;
315:    WORD prev_obj=0;
316:
317:    /* Cursor einschalten */
318:    v_show_c(device,1);
319:
320:    /* Mauszeiger innerhalb des Submenus? */
321:    while ((mx>=sub_menu[sm_index].ob_x-1) &&
322:           (mx<=sub_menu[sm_index].ob_x+
            sub_menu[sm_index].ob_width) &&
323:           (my>=sub_menu[sm_index].ob_y) &&
324:           (my<=sub_menu[sm_index].ob_y+
            sub_menu[sm_index].ob_height+1))
325:    {
326:      /* Eintrag unter Mauszeiger suchen */
327:      index=sub_menu[sm_index].ob_head;
328:      while (sub_menu[index].ob_type==G_STRING)
329:        if ((my>sub_menu[sm_index].ob_y+
              sub_menu[index].ob_y) &&
330:            (my<sub_menu[sm_index].ob_y+
              sub_menu[index].ob_y+
331:              sub_menu[index].ob_height))
332:          break;
333:        else
334:          index=sub_menu[index].ob_next;
335:
336:      /* gefunden? */
337:      if (index!=sm_index)  /* ja */
338:      {
339:        /* Eintrag ungleich letztem angewählten
              Eintrag? */
340:        if (index!=prev_obj)      /* ja */
341:        {
342:          /* zum Zeichnen Mauszeiger ausschalten */
343:          v_hide_c(device);
344:
345:          /* vorigen Eintrag normalisieren */
346:          if (prev_obj && !(sub_menu[prev_obj]
              .ob_state & DISABLED))
347:            switch_entry(FALSE);
348:          /* neuen Eintrag selektieren, falls
              möglich */
349:          pxyarray[0]=sub_menu[sm_index].ob_x+1;
350:          pxyarray[1]=sub_menu[sm_index].ob_y+
                        sub_menu[index].ob_y;
351:          pxyarray[2]=pxyarray[0]+
                        sub_menu[index].ob_width-1;
352:          pxyarray[3]=pxyarray[1]+
                        sub_menu[index].ob_height-1;
353:          switch_entry(sub_menu[index].ob_state &
          DISABLED);
354:
355:          v_show_c(device,1);
356:          prev_obj=index;
357:        }
358:
359:        /* Maustaste gedrückt? */
360:        if (mbutton)    /* ja */
361:        {
362:          /* Menüeintrag anwählbar? */
363:          if (!(sub_menu[index].ob_state &
                DISABLED))  /* ja */
364:            subnum=index; /* Index des Eintrags
                              zurückgeben */
365:          break;
366:        }
367:      }
368:      /* aktuelle Mauskoordinaten holen */
369:      vq_mouse(device,&mbutton,&mx,&my);
370:    }
371:    /* Mauszeiger wieder ausschalten */
372:    v_hide_c(device);
373:  }
```

SUBMENU.C

```
 1:  /**********************************************/
 2:  /*              Submenü-Demo V1.14            */
 3:  /*     - Einsatz von Submenüs unter GEM -     */
 4:  /*     - lauffähig unter allen TOS-Versionen
             und in allen Auflösungen -           */
 5:  /*       (in dieser Fassung für Megamax-C)    */
 6:  /*       by Uwe Hax    (c) MAXON Computer GmbH */
 7:  /**********************************************/
 8:
 9:
10:  #include <gemdefs.h>
11:  #include <obdefs.h>
12:  #include <osbind.h>
13:  #include <portab.h>
```

→

```
14:
15:  WORD contrl[12];
16:  WORD intin[128];
17:  WORD ptsin[128];
18:  WORD intout[128];
19:  WORD ptsout[128];
20:  WORD int_in[11];
21:  WORD int_out[57];
22:
23:  WORD handle;
24:  WORD dummy;
25:  EXTERN WORD global[];
26:
27:  /**********************************************/
28:  #define MAX_SUBMENU    4  /* 4 Submenüs */
29:
30:  #include "menu.h"        /* RSC-Definitionen
                                für Menüs */
31:  #include "submenu.h"     /* RSC-Definitionen
                                für Submenüs */
32:
33:  EXTERN WORD subnum;
     /* Rückgabewert: angeklickter Submenüeintrag */
34:  /**********************************************/
35:
36:
37:
38:  main()
39:  {
40:    WORD mesg_buff[8];
41:    OBJECT *menu,*submenu;
42:    WORD event;
43:    OBJECT **rcs_pointer;
44:    OBJECT *rcs_adr;
45:    WORD ende=FALSE;
46:    char string[20];
47:    WORD m_index[MAX_SUBMENU];
48:
49:    /* Programm anmelden */
50:    appl_init();
51:    handle=graf_handle(&dummy,&dummy,&dummy,
                         &dummy);
52:    open_vwork();
53:
54:    /* Resource-Files laden */
55:    /* 1. Resource-File mit den Submenüs */
56:    if (!rsrc_load("submenu.rsc"))
57:    {
58:      form_alert(1,"[3][SUBMENU.RSC nicht
                 gefunden!][ Abbruch ]");
59:      exit();
60:    }
61:    rsrc_gaddr(R_TREE,MENU,&submenu);
62:
63:    /* Adresse des Resource-Files merken */
64:    rcs_pointer=(OBJECT **)&global[5];
65:    rcs_adr=*rcs_pointer;
66:
67:    /* 2. Resource-File mit allen anderen
          Definitionen: normale Menüs,
68:       Dialogboxen, ... laden */
69:    if (!rsrc_load("menu.rsc"))
70:    {
71:      form_alert(1,"[3][MENU.RSC nicht gefunden!]
                 [Abbruch ]");
72:      exit();
73:    }
74:    rsrc_gaddr(R_TREE,MENU,&menu);
75:
76:    /* Submenüs installieren */
77:    m_index[0]=BILDTYP;  /* Indizes aller Menü-
                              einträge, die Titel */
78:    m_index[1]=SYSTEM;   /* f.Submenüs sein    */
79:    m_index[2]=COMPUTER; /* sollen,in fortlaufender
                              Reihenfolge */
80:    m_index[3]=STIL;
81:    init_submenus(handle,menu,submenu,MAX_SUBMENU,
                     m_index);
82:
83:    graf_mouse(ARROW,&dummy);
84:    menu_bar(menu,TRUE);
85:
86:    do
87:    {
88:      event=evnt_multi(MU_MESAG | MU_TIMER,dummy,
                          dummy,dummy,dummy,dummy,
89:                       dummy,dummy,dummy,dummy,
                          dummy,dummy,dummy,dummy,
                          mesg_buff,100,
90:                       0,&dummy,&dummy,&dummy,
                          &dummy,&dummy,&dummy);
91:
92:      if (event & MU_MESAG)
93:      {
94:        if (mesg_buff[0]==MN_SELECTED)
95:        {
96:          switch (mesg_buff[4])
97:          {
98:            case BILDTYP:
99:            case SYSTEM:
100:           case COMPUTER:
101:           case STIL:       redraw_bg();
102:                            break;
              /* bei Anklicken der Submenütitel
                 müssen auch die Submenüs
103:             einklappen */
104:           case ENDE:       ende=TRUE;
105:                            break;
106:         }
107:         menu_tnormal(menu,mesg_buff[3],TRUE);
108:       }
109:     }
110:
111:
112:     if (event & MU_TIMER) /* subnum muß
                   regelmäßig abgefragt werden */
113:     {        /* da kein MU_MESAG gesendet wird */
114:       if (subnum)  /* Submenü angewählt? */
115:       {
116:         /* Haken setzen bzw. löschen */
117:         if (submenu[subnum].ob_state & CHECKED)
118:           menu_icheck(submenu,subnum,FALSE);
119:         else
120:           menu_icheck(submenu,subnum,TRUE);
121:
122:         /* Indexnr. des angewählten Submenüs
               ausgeben */
123:         sprintf(string,"Submenu-Index: %d ",
                     subnum);
124:         v_gtext(handle,0,150,string);
125:         subnum=0;      /* anschließend
                              zurücksetzen */
126:       }
127:     }
128:   }
129:   while (!ende);
130:
131:   /* Resource-Files wieder freigeben */
132:   rsrc_free();
133:   *rcs_pointer=rcs_adr;
134:   rsrc_free();
135:   v_clsvwk(handle);
136:   appl_exit();
137: }
138:
139: open_vwork()
140: {
141:   WORD i;
142:
143:   for (i=1; i<10; i++)
144:     int_in[i]=1;
145:   int_in[10]=2;
146:   v_opnvwk(int_in,&handle,int_out);
147: }
```

MENUDEMO.C

Anwendungen in dBMAN

Datumsfunktionen

Seit gut einem Jahr ist nun die Version 5.01 des Datenbanksystems dBMAN 5.01 auf dem Markt. Mit der dazugehörigen Programmiersprache und dem Greased Lightning-Compiler steht für den ATARI ST ein Entwicklungspakket zur Verfügung, das hinsichtlich der Geschwindigkeit verblüffende Ergebnisse zeigt, aber auch bei ganz speziellen Aufgabenstellungen kaum noch Wünsche offen läßt. Mit einigen Kniffen kommt man aber auch in diesen Fällen oft recht schnell und einfach zu brauchbaren Ergebnissen.

Dabei ist interessant, daß dBASE III plus-Anwendungen auf dem dBMAN-Interpreter relativ problemlos laufen. Umgekehrt ist das nicht ganz so einfach, da dBMAN sowohl in der Behandlung von Variablen mehr Möglichkeiten zur Verfügung stellt als auch eine wesentlich größere Funktionsvielfalt als dBASE III plus besitzt. Dies gilt insbesondere für den Bereich der Menügestaltung. Aber auch die Funktionen zur String-Selektion und für die Behandlung von Zeit- und Datumsangaben erleichtern die Programmierung erheblich.

Datenbanken...

... mehr als nur Adressen- und Artikelverwaltung. Diese Kursreihe wird auf diese und andere, etwas "exotischere" Funktionen in diesem Datenbanksystem eingehen, und versuchen zu zeigen, was über die eigentliche Behandlung von Datenbanken hinaus möglich ist.

Man kann zwar von grundsätzlichen Regeln für das Programmieren ausgehen, letztendlich aber entwickelt jeder mit der Zeit seinen eigenen individuellen Programmierstil. Die Programme und Programm-Module, die in dieser Reihe vorgestellt und erläutert werden, dienen als Beispiele, an denen prinzipielle Vorgehensweisen deutlich gemacht werden sollen. Dabei orientiere ich mich an folgenden Leitlinien:

a.) Die Programm-Module sollen möglichst kompakt sein.
b.) Die Programm-Module sollen flexibel einsetzbar sein.
c.) Die Programm-Module sollen möglichst kurz sein.
d.) Das Programm als Ganzes soll möglichst schnell sein
e.) und daher häufige Disketten-/Plattenzugriffe vermeiden.

Die Realisierung insbesondere der ersten beiden Leitlinen führt aber doch leicht zur Unübersichtlichkeit, so daß ich es für sinnvoller gehalten habe, die Programme nur mit Minimal-kommentaren zu versehen. Bei der Beschreibung der Anwendungsmöglichkeiten, des Aufbaus und der Funktionsweise des Programms sowie der Erklärung und Kommentierung wichtiger Befehle, Funktionen und Ausdrücke werde ich auf die jeweiligen Zeilennummern im entsprechenden Listing verweisen.

In diesem ersten Teil wollen wir im wesentlichen auf den Umgang mit *Datumsfunktionen* eingehen. Ein kleines, aber sehr nützliches Programm soll als Anschauungsbeispiel dienen. Zunächst mal Grundsätzliches zu den Datumsfunktionen.

Sowohl Memory- als auch Feldvariablen können Daten im Datumsformat aufnehmen. Das Systemdatum *DATE()* ist das einzige Datum, das bereits im richtigen Format vorliegt (ich gehe im folgenden davon aus, daß dies immer das aktuelle Datum ist.).

Datumsanzeige

Mit den Befehlen *SET DATE TO GERMAN* und *SET CENTURY ON* oder *ASSIGN LONGYEAR(T)* erfolgt die Anzeige von Variablen, die als *DATUM* definiert sind, im Format *tt.mm.jjjj*.

Rechnen im Datumsformat

Die Rechenoperationen mit Datumsvariablen beschränken sich logischerweise auf Addition und Subtraktion. Das Ergebnis ist entweder wieder ein Datum oder ein numerischer Wert.

```
DATUM +/- num. Wert = DATUM
DATUM +/- DATUM = num. Wert
```

Datumsumformung

Es gibt nun eine ganze Reihe von Umformungsmöglichkeiten, die ebenfalls für die Anzeige und Berechnung notwendig sind. Um überhaupt ein Datum, das noch nicht existiert, in eine Variable schreiben zu können, muß es als Character-String vorliegen und kann dann beliebig umgeformt werden.

```
CTOD('01.05.1989')   C(haracter) TO D(ate)
DTOC(Y.DATUM)        D(ate) TO C(haracter)
```

Datumseinheiten extrahieren

Als numerischer Wert können der Tag, der Monat, das Jahr und die laufende Nummer des Wochentags ermittelt werden.

```
DAY(DATE())
MONTH(DATE())
```

YEAR(DATE())
DOW(DATE()) —> D(ay) O(f) W(eek)

Kalender bis 2039

Es gibt eine ganze Reihe von Anwendungen, bei denen es ganz sinnvoll wäre, wenn das Programm selbständig erkennen könnte, ob ein bestimmter Tag ein (arbeitsfreier) Feiertag ist. Man denke zum Beispiel an die Datenbank einer Autovermietung, die natürlich den Fahrzeugpark mit Kundendienstterminen, Fälligkeitsdaten der Versicherungsprämien sowie die Kundendatei verwaltet und Rechnungen schreibt.

Ein anwenderfreundliches Programm-Modul *RECHNUNG SCHREIBEN* müßte Sonn- und Feiertage erkennen können, um dann die entprechend höheren Tarife für die Berechnung der Ausleihgebühr zugrundezulegen. Aber auch Anwenderprogramme für Schulen/Internate etc. können, wenn die Feiertage und das Bundesland definiert sind, Ferien berechnen oder dafür sorgen, daß Prüfungstermine o.ä. nicht versehentlich auf einen schulfreien Tag gelegt werden. Ich selbst bin durch die Verwaltung meiner Kurse auf die Notwendigkeit einer Feiertagserkennung innerhalb einer Datenbankanwendung gekommen.

Das Programm

Es geht also darum, ein bestimmtes gegebenes Datum daraufhin zu überprüfen, ob es auf ein Wochenende oder auf einen gesetzlichen Feiertag fällt. Das Programm *ISTFEI.CMD* verlangt die Eingabe eines Datums. Dieses Datum wird nur aufgrund der Jahreszahl daraufhin überprüft, ob der Rechenbereich von dBMAN nicht überschritten wird. Es folgt die Ausgabe des Wochentags und des Monats in Worten. Anschließend werden die einzelnen Feiertage berechnet, in numerische Werte umgewandelt und ein Vergleich mit dem eingegebenen Datum durchgeführt. Das Ergebnis des Vergleiches ist 0, wenn kein Feiertag vorliegt, oder nimmt einen Wert zwischen 1 und 15 an. Dieser Wert ist identisch mit der Position des gefundenen Feiertags in einer Liste der 15 möglichen Feiertage. Der entsprechende Feiertag wird als C-String einer Variablen übergeben. Die DO WHILE-Schleife wird wiederholt, solange die Eingabe nicht mit der Taste >ESC< beendet wird.

Tage und Monate selbst gemacht

Da dBMAN nur in englischer Version vorliegt, erfolgt die Darstellung von Wochentagen und Monaten in Worten ebenfalls in englischer Sprache also *CDOW('11/09/89')* ergibt *Monday*, und *CMONTH('15/05/89')* ergibt *May*.

Ein kleiner Trick hilft weiter. Am besten definieren wir am Anfang jeder Anwendung zwei Variablen, die alle Wochentage und Monate enthalten sowie die Variable *X.DAT* für das jeweilige Datum, dessen Monat oder Wochentag angezeigt werden soll (Listing Zeile 6-8).

Die Funktion *EXTRACT()* sucht nun den durch *DOW(datum)* errechneten numerischen Wochentag aus der durch Kommas getrennten Wochentagsliste heraus. Das "Trennzeichen" (delimiter) kann frei gewählt werden. Dabei geht es nicht nur darum, welches Zeichen schöner oder übersichtlicher ist, sondern vielmehr erleichtert es die Behandlung von Ausdrükken mit fest vorgegebenen Trennzeichen wie zum Beispiel Dateisuchpfade (Backslash "\").

EXTRACT('delimiter',LISTE,n'ter)

Die "Schwester" dieser Funktion erlaubt die Wahl von zwei Trennzeichen.

EXTRACT2(delimiter1,delimiter2,C-String,n)

Allerdings darf "n" hierbei nur Werte zwischen 1 und 3 annehmen, entsprechend wird dann links, rechts oder zwischen beiden Trennzeichen extrahiert. Das vorangestellte Makrosymbol "&", (Listing Zeile 24,25) bewirkt, daß bei der Ausführung der Variablenname *X.WOT* und das Symbol selbst durch den Inhalt der Variablen ersetzt und die *EXTRACT*-Funktion ausgeführt werden. Wir werden im zweiten Teil dieser Reihe noch ausführlich auf die Verwendung von Makros eingehen.

Feiertage Das ganze Jahr

Da Feiertage nicht jedes Jahr neu und beliebig festgesetzt werden, sondern ganz bestimmten Regeln folgen, können sie berechnet werden. Grundsätzlich gibt es drei unterschiedliche Arten von Feiertagen:

Bei den sogenannten "fixen Feiertagen" wie dem 1.Mai, dem 17.Juni oder dem 24.Dezember ändert sich ja nur der Wochentag. Das Datum selbst ist konstant und stellt somit kein Problem dar. Man braucht nur das entsprechende Jahr als C-String anzuhängen (Listing Zeile 49-56). Schwieriger wird es bei den beweglichen Feiertagen.

Wann ist eigentlich Ostern 2002?

Die christlichen Feste und damit die meisten gesetzlichen Feiertage richten sich nach dem Ostersonntag. Der wiederum ist aber bezüglich seines "Termins", "heidnischen" Ursprungs und richtet sich deshalb nach dem Mondzyklus. So hat das Konzil von Nicäa im Jahr 325 n.Chr. beschlossen, daß Ostern immer auf den ersten Sonntag nach dem ersten Frühlingsvollmond fällt.

Den Algorithmus, mit dem der Ostersonntag letzendlich berechnet wird, ausführlich zu erklären, würde den Rahmen dieses Kurses sprengen. Grundsätzliches über die Kalenderberechnung ist aber sehr interessant und soll deshalb auch kurz erwähnt werden.

Wie jeder weiß, ist das sogenannte "bürgerliche Jahr" in 365 Tage bzw. 12 Monate unterteilt. Die Monate hängen ursprünglich, wie der Name schon vermuten läßt, mit dem Lichtwechsel des Mondes zusammen. Da die Zeit zwischen zwei Neumonden nur etwa 29,53 mittlere Sonnentage umfaßt, müßte in einem reinen Mondkalender von Zeit zu Zeit ein "Schaltmonat" eingefügt werden, um zu verhindern, daß der (Mond-)Kalender zu sehr von den Jahreszeiten abweicht.

29,53 Tage * 12 Monate = 354,36 Tage/Jahr

365 Tage/Jahr - 354 Tage/Jahr	~ 9 Tage
(bürgerl.Jahr) - (Mondjahr)	Abweichung pro Jahr.

Das entspricht schon innerhalb von 10 Jahren einer Verschiebung von drei Monaten, das hieße zum Beispiel, daß im März schon Sommer wäre.

Nach mehreren Kalenderreformen, bei denen teilweise tatsächlich mehrere Monate einfach übersprungen wurden, führten dann endlich die Römer eine Monatsdauer von 30, 31 bzw. 28 Tagen ein. Dummerweise führt aber auch diese Einteilung alle vier Jahre zu einer Abweichung von einem Tag, die durch den berühmten 29. Februar des Schaltjahres korrigiert wird. Dadurch haben wir nur noch alle 100 Jahre ungefähr ein Tag zuviel. Deshalb haben diejenigen, die an

diesem denkwürdigen Tag geboren sind, alle 1000 Jahre doppelt Pech, weil in den Jahren, die ohne Rest durch 100 teilbar sind, das Schaltjahr, der 29. Februar und die Geburtstagsfete ausfallen.

Doch um Schaltjahre und andere Unregelmäßigkeiten brauchen wir uns nicht zu kümmern, das erledigt dBMAN automatisch für uns. Allerdings nur bis zum 31.12.2039. Daher die obere Jahresbegrenzung (Listing Zeile 14/15). Zurück zu unserer Ostersonntag-Berechnung. Frühlingsanfang ist gewöhnlich am 21.März, in Schaltjahren entsprechend einen Tag früher. Das ist genau an dem Tag des Jahres, an dem die Sonne den Himmelsäquator von Süden nach Norden überschreitet. Auf diesem Hintergrund hat C.F. Gauss (1777-1855) einen Algorithmus entwickelt, mit dem der Ostersonntag berechnet werden kann (Listing Zeile 18-32). Dabei bedeuten

Y.JAHR	die Jahreszahl
Y.D und Y.E	Divisionsreste (sie berechnen Schaltjahre und Wochentage)

Die Ermittlung der Divisionsreste erfolgt mit der Funktion *MOD(wert1,wert2)*. Sie liefert den Rest der Division von *Wert1* und *Wert2*. Bei dBASE II hätte man noch schreiben müssen:

```
MOD(wert1,wert2) = wert1 - INT(wert1/
                   wert2)*wert2
```

Je nachdem, ob das Jahr zwischen 1900 und 2099 oder 2200 und 2199 liegt, muß bei der Berechnung von *Y.E* der Wert 5 oder 6 eingesetzt werden (Listing Zeile 31).

Nützlich ist hier die Funktion *IIFN()*, die manchem vielleicht aus Tabellenkalkulationen bekannt vorkommt. [Meist lautet sie dort *WENN()*.] Sie hilft, so manche Programmzeile einzusparen, zumal sie in fünf Variationen existiert.

```
IIF(); IIFC(); IIFD(); IIFN(); IIFL()
```

Die Syntax ist für alle Formen identisch:

```
IIF(Bedingung,Ausgabe wenn t,
    Ausgabe wenn f)
```

Die einfache Form von *IIF()* kann zwei unterschiedliche Ausgabetypen besitzen. Die Erweiterungen *IIFC,D,N,L* stehen jeweils für den bestimmten Ausgabetyp (Character, Datum, numerisch, logisch). Da dBMAN bei der Eingabe eines Datums, das größer als der 31.12.2039 ist, ohnehin unerbittlich eine Fehlermeldung auswirft, kann hier auch 5 als fester Wert eingesetzt werden.

```
 1: * ISTFEI.CMD
 2:
 3: ***** PARAMATER / GLOBALE VARIABLE
 4: SET TALK OFF
 5: SET DB3 ON
 6: X.DAT = DATE()
 7: X.MO = 'EXTRACT(",","Januar,Februar,März,
           April,Mai,Juni,Juli,August,September,
           Oktober,November,Dezember",MONTH(X.DAT))'
 8: X.WOT ='EXTRACT(",","Sonntag,Montag,Dienstag,
           Mittwoch,Donnerstag,Freitag,Samstag",
           DOW(X.DAT))'
 9:
10: DO WHILE .T.
11: ERASE
12: ***** EINGABE
13: X.SDAT = DATE()
14: @ 2,5 SAY 'GESUCHTER TAG: ' GET X.SDAT PICT
      '##.##.####' VALID NRANGE(YEAR(X.SDAT),
      1989,2039) ;
15: ERRMSG 'DATUM AUSSERHALB DES DERZEITIGEN
           RECHENBEREICHES (1989-2035)'
16: @ 21,1 SAY'ENDE MIT  ESC'
17: @ 3,0 TO 3,80
18: READ
19: ***** EXIT BEDINGUNG
20:      IF LASTKEY() = 27
21:          RETURN
22:      ENDIF
23: X.DAT = X.SDAT
24: @ ROW()+1,5 SAY &X.MO
25: @ ROW()+1,5 SAY &X.WOT
26:
27: ***** BERECHNUNG DES JEWEILIGEN OSTERDATUMS
28: Y.JAHR =YEAR(X.SDAT)
29: Y.VAR = IIFN(Y.JAHR-1989=0,1,((Y.JAHR-1989)*5)+1)
30: Y.D = MOD(19*MOD(Y.JAHR,19)+24,30)
31: Y.E = MOD(2*MOD(Y.JAHR,4)+4*MOD(Y.JAHR,7)+6*
          Y.D+IIFN(NRANGE(Y.JAHR,1900,2035),5,6),7)
32: X.DOST = CTOD(STR(22+Y.D+Y.E,2,0)+'.03.'+
           STR(Y.JAHR,4,0))
33:
34: ***** ASCHERMITTWOCH/LISTE/REFERENZDATUM
35: Y.DAMI=X.DOST-46
36: Y.FLISTE='ROSENMONTAG/KARFREITAG/OSTERMONTAG/
           CHR.HIMMELF./PFINGSTMONTAG/;
37:        FRONLEICHNAM/3 KÖNIG/TAG DER ARBEIT/
           17.JUNI/MARIA HIMMELF./;
38:        ALLERHEILIGEN/HL.ABEND/1.FEIERT./
           2.FEIERT./BUP+BETTAG'
39: Y.J1=CTOD('01.01.'+STR(Y.JAHR,4,0))
40:
41: ***** BERECHNUNG DER FEIERTAGE FÜR DAS GANZE JAHR
42: Y.ND0=X.SDAT-Y.J1
43: Y.ND1=Y.DAMI-2-Y.J1
44: Y.ND2=Y.DAMI+44-Y.J1
45: Y.ND3=Y.DAMI+47-Y.J1
46: Y.ND4=Y.DAMI+85-Y.J1
47: Y.ND5=Y.DAMI+96-Y.J1
48: Y.ND6=Y.DAMI+106-Y.J1
49: Y.ND7=CTOD('07.01.'+STR(Y.JAHR,4,0))-Y.J1
50: Y.ND8=CTOD('01.05.'+STR(Y.JAHR,4,0))-Y.J1
51: Y.ND9=CTOD('17.06.'+STR(Y.JAHR,4,0))-Y.J1
52: Y.ND10=CTOD('15.08.'+STR(Y.JAHR,4,0))-Y.J1
53: Y.ND11=CTOD('01.11.'+STR(Y.JAHR,4,0))-Y.J1
54: Y.ND12=CTOD('24.12.'+STR(Y.JAHR,4,0))-Y.J1
55: Y.ND13=CTOD('25.12.'+STR(Y.JAHR,4,0))-Y.J1
56: Y.ND14=CTOD('26.12.'+STR(Y.JAHR,4,0))-Y.J1
57: Y.ND15=(Y.J1+46*7-DOW(Y.J1)+4)-Y.J1
58:
59: Y.NLISTE=NLIST(Y.ND0,Y.ND1,Y.ND2,Y.ND3,Y.ND4,
           Y.ND5,Y.ND6,Y.ND7,Y.ND8,Y.ND9,Y.ND10,
           Y.ND11,Y.ND12,Y.ND13,Y.ND14,Y.ND15)
60: X.FEIERT=EXTRACT('/',Y.FLISTE,Y.NLISTE)
61: @ ROW()+2,5 SAY IIFC(Y.NLISTE<>0,X.FEIERT,'')
62: WAIT
63: ENDDO
64: RETURN
```

Der Ostersonntag ist dann der *(Y.D+Y.E+22)*'te März des entsprechenden Jahres. Bekanntlich ist aber Ostern viel öfter im April als im März, so daß der Ausdruck *(Y.D+Y.E+22)* zwangsläufig einen Wert annehmen muß, der größer als 31 ist. Normalerweise beschert uns dBMAN in so einem Fall den Fehler

ERMSG->Invalid date, bzw. wir können das "GET-Feld" nicht verlassen, solange wir kein gültiges Datum eingegeben haben. Dabei ist als einzige Ausnahme zu beachten, daß ein "leeres" Datum *CTOD(' . . ')* akzeptiert wird. Durch den Befehl *SET DB3 ON* (Listing Zeile 5) wird dies verhindert und ein falsches Datum in ein gültiges Datum umgerechnet. So wird aus dem 33. März automatisch der 2. April. Bei Daten, die aus einem gültigen Datum berechnet worden sind, ist das äußerst praktisch, bei Benutzereingaben allerdings werden auch Tippfehler klaglos akzeptiert.

Dieser Befehl hat allerdings noch andere, u.U. nicht erwünschte Effekte. DBF- und NDX-Files werden dann im dBASE III plus-Format angelegt, und der Befehl *CLEAR ALL* selektiert den Arbeitsbereich *A (FJ)* anstatt *G (FP)[rimary]*. Also, entweder gleich wieder ausschalten oder selbst umformen, z.B.

```
IF= 22+Y.E+Y.D >31
X.DOST=(Y.E+Y.D-9)'ter April
ELSE ....
```

und die Eingabe mit dem GET-Parameter *VALID* wie z.B im Listing, Zeile 17 überprüfen. dBMAN besitzt noch eine ganze Reihe weiterer Möglichkeiten, Benutzereingaben auf ihre "Sinnhaftigkeit" hin zu überprüfen. Wir werden in den nächsten Folgen ausführlich darauf eingehen.

Nun könnten die restlichen Feste, die sich auf Ostern beziehen, berechnet werden. Da aber der Vergleich nicht einzeln, d.h. mit einer IF- oder CASE-Folge durchgeführt werden soll (das wäre viel zu umständlich), muß das Datum als numerischer Wert vorliegen. Das ist eigentlich nur mit dem sogenannten Julianischen Datum möglich, bei dem die Tage, ohne Berücksichtung der Jahre oder Monate, einfach vom 01.01.4713 v.Chr. durchgezählt werden.

dBMAN rechnet zwar intern mit dieser Datums-Seriennummer, stellt sie aber nicht direkt als numerischen Wert zur Verfügung, so daß wir uns ein eigenes Referenzdatum erstellen müssen. Wir wählen für *Y.J1* den 01.Jan. des jeweiligen Jahres, damit die Zahlen nicht zu groß werden, und berechnen den n-ten Tag dieses Jahres (Listing Zeile 42-56). Wenn man das Programm benutzen will, um eine Reihe von Terminen nacheinander zu berechnen, die u.U. einen Jahreswechsel beinhalten, ist es günstiger, ein konstantes Referenzdatum zu wählen und den Eingabebereich entsprechend nach unten zu begrenzen, um keine negativen Referenzen zu erhalten.

Da auch der Rosenmontag in unserer Kennung berücksichtigt werden soll, ist es übersichtlicher, zur Berechnung der weiteren Feiertage nicht Ostern zugrundezulegen, sondern den Aschermittwoch. An diesem Tag beginnt die sogenannte fünfundvierzigtägige Fastenzeit, so daß gilt:

```
Aschermittwoch = Ostern-46
Rosenmontag    = Aschermittwoch-2
Pfingsten      = Aschermittwoch+85
usw.
```

Zunächst aber wird noch die Variable *FLISTE* definiert (Listing Zeile 36-38). Sie beinhaltet die Namen der Feiertage in derselben Reihenfolge, wie sie den Variablen *Y.ND1 - Y.ND15* zugeordnet werden. Die Variable *Y.ND0* enthält das zu suchende Datum. Dabei sind *Y.ND1-Y.ND6* von Ostern abhängige Feste und *Y.ND7-Y.ND14* fixe Feiertage. *Y.ND15* berechnet den Buß- und Bettag (Listung Zeile 57), der zu der dritten Gruppe der Feiertage gehört. Er ist immer am 3. Mittwoch im November (46.Woche). Falls der 1. Nov. selbst ein Mittwoch ist, zählt er nicht mit.

Termine, am n-ten Montag, Dienstag... eines bestimmten Monats können mit der *DOW(datum)* berechnet werden. *DOW (DATUM)* liefert den numerischen Wert des Wochentages:

```
Sonntag  = 1 Montag  = 2......
Samstag = 7
```

Der Ausdruck *-(DOW(datum))+4)* ergibt also immer einen Wert zwischen

```
+3= -1+4 (für Datum = Sonntag)
- 3= -7+4 (für Datum = Samstag)
```

und wird zu 0, wenn das Datum auf den entsprechenden Wochentag fällt (Listing Zeile 57). Die bayrischen Sommerferien z.B. beginnen in der Regel am letzten Donnerstag im Juli. Wenn wir das Datum *Y.FANF* des 1. Ferientages im Jahr XY suchen, so gilt Donnerstag ist der 5. Tag der Woche.

```
DOW(donnerstag)= 5

Y.JAHR= XY
Y.VAR1=  CTOD('31.07.'+STR(Y.JAHR,4,0))
Y.FANF=  IIFD(DOW(Y.VAR1)<5,Y.VAR1-
                (DOW(Y.VAR1)+2),Y.VAR1+
                (-DOW(Y.VAR1)+5))
```

Wenn also der 31.07. nicht größer als Donnerstag ist, wird der Ausdruck

```
-(DOW(Y.VAR1)+2)
```

zu 0, sonst wird

```
-(DOW(Y.VAR1)+5))
```

zu 0 und die Bedingung

```
Ferienanfang = <= 31.07.XY
Ferienanfang = >= 31.07.XY -7
```

ist immer erfüllt.

Im Prinzip kann so jedes Datum, das für eine bestimmte Anwendung relevant ist, berechnet und in die Liste eingefügt werden. Eine Grenze stellt lediglich die maximal erlaubte Zeilenlänge dar, aber wer will, kann ja eine zweite Liste erstellen.

Wenn so alle Tage des Jahres, die in Frage kommen, bereitstehen, kann die *NLIST(suchwert,wert1,wert2...wertn)* nun die Position des Wertes in der Liste, der dem Suchwert entspricht, in die Variable *Y.NLISTE* schreiben (Listing Zeile 59). Ist kein identischer Wert vorhanden, wird *Y.NLISTE* zu 0.

Die Funktion *EXTRACT()* extrahiert mit Hilfe des Wertes von *Y.NLISTE* den entsprechenden String aus *Y.FLISTE* (Listing Zeile 60). Der Name des Feiertags steht nun in der Variablen *X.FEIERT*. Die Funktion *IIFC()* benötigen wir, weil *EXTRACT()* (leider) keinen Null-String liefert, wenn *Y.NLISTE (Position) = 0* ist (Listing Zeile 62).

Das abgedruckte Listing von *ISTFEI.CMD* ist hier (zum Austesten), was die Eingabemöglichkeit und die Verwendung von Memory-Variablen anbelangt, als eigenständig lauffähiges Programm geschrieben. Zur Anzeige liegen daher globale X.-Variablen vor.

X.SUFEI	das überprüfte Datum im Datumsformat
&X.WOT	der Wochentag in Worten
&X.MO	der Monat in Worten
X.FEIERT	der Name des Feiertags

Kommen wir aber noch einmal zu dem möglichen Anwendungsbereich als Programm-Modul beim Programmteil *RECHNUNG SCHREIBEN* in der Datenbank einer Autovermietung zurück. Ein vereinfachter Ablaufplan könnte folgendermaßen aussehen:

Der Programmteil *RECHNUNG SCHREIBEN* übernimmt aus dem KUNDEN-Datensatz das Datum und aus dem Fahrzeugpark-Datensatz den Tarif. Nun muß festgestellt werden, ob das fragliche Datum ein Wochenende oder Feiertag ist.

Wenn *DOW(Ausleihtag)* 1 oder 7, also ein Wochenende ist, kann man sich ja den "Ausflug" nach *ISTFEI.CMD* sparen. Ansonsten muß das Datum nach *ISTFEI.CMD* (Teil 5-7) übermittelt und der Name des Feiertages als Variable wieder nach *RECHNUNG SCHREIBEN* übernommen werden. Wenn nun ein Feiertag erkannt worden ist,

```
FEIERT <> ''
```

wird der Tarif z.B. um 25% erhöht, und es gilt *TARIF = TARIF*1.25*, wenn der Programmteil *RECHNUNG SCHREIBEN* fortgesetzt wird.

In solch einem Fall bietet sich ein anderer Umgang mit den Variablen an. Wie schon gesagt, stellt dBMAN wesentlich mehr Möglichkeiten als dBASE III plus zur Behandlung von Memory-Variablen zur Verfügung.

Memory-Variablen-Handling

In der Standardkonfiguration von dBMAN können gleichzeitig 256 Variablen ohne Präfix und 128 X.-Variablen als globale Variablen und 64 Y.-Variablen bzw. Z.-Variablen pro Programm als lokale Variablen aktiv sein. Dabei entsprechen die 256 Variablen ohne Präfix den *PRIVAT/PUBLIC*-Variablen von dBASE III plus, d.h sie gelten als *PRIVAT* und werden automatisch nach Beendigung des Programms, das sie definiert hat, gelöscht, wenn sie nicht ausdrücklich vor der Belegung mit Werten als *PUBLIC* definiert wurden. Das bedeutet:

- Eine Variable darf nur einmal zu *PUBLIC* erklärt werden.
- Die Umwandlung von *PRIVAT* in *PUBLIC* ist nicht möglich.
- Die Umwandlung von *PUBLIC* in *PRIVAT* erzeugt eine neue Variable gleichen Namens.

PRIVAT-Variablen können aber auch als *PARAMETER* mit dem Befehl *DO programm WITH PARAMETER var1, var2,...* an andere Programme übergeben werden. Wir werden diese Möglichkeit im zweiten Teil noch näher kennenlernen. Auf X.-Variablen haben alle Programme uneingeschränkt Zugriff. Sie können wie *PUBLIC*-Variablen nur mit dem Befehl *RELEASE [FX] ALL [EXEPT/LIKE maske]* gelöscht werden.

Y.- und Z.-Variablen werden wie *PRIVATE* gelöscht, nur mit dem Unterschied, daß sie in einem mit *DO program.cmd* aufgerufenen Programm als Z.- und Y.-Variablen weiterhin verwendbar und veränderbar sind und nach der Rückkehr in das Programm, in dem sie ursprünglich definiert worden sind, mit neuem Inhalt als Y.- bzw. Z.-Variablen weiterverarbeitet werden können. Erst nach Verlassen des Ursprungsprogramms werden sie endgültig gelöscht. Anschaulich - aber vereinfacht - könnte man sagen, für jede Y.-Variable existiert im folgenden Programm-Modul eine Z.-Variable. Beim Rücksprung verhält es sich genau umgekehrt. Konkret würde sich also für die Einbindung des Programm-Moduls in die Rechnungsschreibung einer Autovermietung dann folgende Variablenbenennung anbieten.

WOT, MO und *DAT* werden im Hauptprogramm vor der ersten DO WHILE-Schleife als *PUBLIC* definiert und dann erst einmal mit *DATE()* belegt.

In der Datei mit dem *ALIAS*-Namen "KUNDEN" stehen der oder die Ausleihtage im 3. Feld des jeweiligen Datensatzes der Form *tt.mm.jjjjnnn* zur Verfügung. *nnn* steht für die Anzahl der Ausleihtage. Im Programm-Modul *RECHNUNG SCHREIBEN* werden die Variable für den ersten Ausleihtag als

```
Y.SUFEI = CTOD($(EXTRACT(',',FIELD
          (%KUNDEN,3),1),1,10))
```

die Anzahl der Tage als

```
Y.ANZ  = VAL($(EXTRACT(',',FIELD
         (%KUNDEN,3),1),11,3))
```

definiert, so daß im Modul *ISTFEI.CMD* die Variable *Y.SUFEI* als *Z.SUFEI* (das zu suchende Datum) weiterverarbeitet werden kann. *Y.ANZ*, nunmehr anprechbar als *Z.ANZ*, bestimmt, wie oft das Datum um 1 erhöht und der Vergleich weiter durchgeführt werden muß. Die Variable *FEIERT* wird im erst im Modul *ISTFEI.CMD* und deshalb gleich als *Z.FEIERT* definiert. Nach der Rückkehr zum Programm-Modul *RECHNUNG SCHREIBEN* stehen dann *Y.SUFEI* und *Y.ANZ* wieder und *Y.FEIERT* erstmals als Y.-Variablen zur Verfügung und werden nach Beendigung des Moduls *RECHNUNG SCHREIBEN* automatisch gelöscht.

Diese Möglichkeit gilt also grundsätzlich für verschachtelte Programm-Module. Eine mehrfache Verschachtelung ist natürlich auch möglich. dBMAN erstellt dann beim Aufruf des 3. Programm-Moduls automatisch die Datei *DBMEM.MEM*, so daß bei der Rückkehr immer die entsprechenden Y.- und Z.-Variablen wiedereingelesen werden.

Zur Wahrung der Übersichtlichkeit und in Anbetracht der Diskettenzugriffe, die das Schreiben und Wiedereinlesen von *DBMEM.MEM* erfordert, sollte man im doppelten Sinne des Worte,s aber nicht ohne Not, zu "tief stapeln".

Im nächsten Teil werden wir uns ein komplettes Programm aus mehreren Modulen ansehen, das mit einer Datenbank zusammenarbeitet, in der Tage, Zeiten und die Art der Tätigkeiten, die während des angegebenen Zeitraums ausgeführt wurden, gespeichert sind. Das Beispielprogramm ermöglicht und kontrolliert die Eingabe von neuen Datensätzen, berechnet Stunden, Minuten und erstellt Listen für beliebige Zeiträume, die auf beliebige Ausgabeeinheiten geleitet werden können. Mit einigen Ergänzungen könnte dieses Programm in eine komfortable Arbeitszeit- bzw Arbeitstundenverwaltung für Handwerker oder Freischaffende eingebaut werden.

Peter Neuchel

Numerische Mathematik

Teil 5
Nichtlineare Gleichungen und Nullstellen

In der letzten Folge der Serie Numerische Mathematik behandeln wir nichtlineare Gleichungen und damit auch Verfahren zur Suche von Nullstellen.

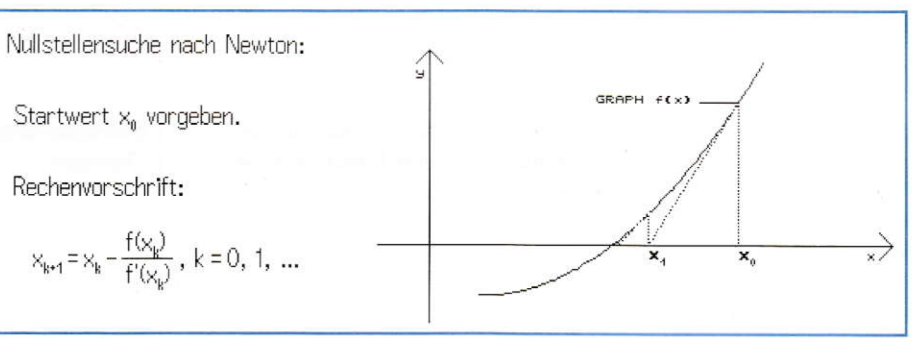


Bild 1

Die Suche nach Nullstellen bzw. Lösungen von nichtlinearen Gleichungen ist nicht immer einfach. Erinnern wir uns an die Schulzeit. Lösungen linearer Gleichungen der Form $ax+b=0$ waren einfach zu bestimmen, bei quadratischen der Form $ax2+bx+c=0$ war dies auch nicht so schwer, womit aber schon die leichten Fälle aufgezählt sind. Lediglich noch für Gleichungen 3. und 4. Ordnung gibt es kompakte Formeln. Lösungen von Gleichungen höherer Ordnung lassen sich formelmäßig nicht mehr angeben.

Bei vielen anderen Gleichungen wie beispielsweise $exp(x)=3$ lassen sich auf mehr oder weniger komplizierte Art Lösungen ermitteln. Wenn diese untereinander gemischt werden - etwa $exp(sin(x))+cos(x)=0$ -, ist es meist gänzlich vorbei. Hier werden wieder numerische Verfahren benötigt.

Betrachten wir zuerst Gleichungen mit einer Unbekannten. Eines der bekanntesten Verfahren zur Bestimmung einer Lösung der Gleichung $f(x)=0$ ist das *Newton-Verfahren* (nach Isaac Newton, geb. 4. 1. 1643 in Woolsthorpe, gest. 31. 3. 1727 in London).

Um das Newton-Verfahren benutzen zu können, muß die Funktion f einigen Bedingungen genügen: f muß in jedem Fall stetig differenzierbar sein. Andere Voraussetzungen wollen wir hier nicht beachten. Spätestens wenn das Verfahren nicht konvergiert, d. h. keine Lösung gefunden werden kann, sollte man vermuten, daß eventuell gewisse Voraussetzungen eine entscheidende Rolle spielen.

Da f als stetig differenzierbar vorausgesetzt wurde, darf man die Funktion ableiten. In Bild 1 ist die Iterationsvorschrift zu sehen. Etwas ist in jedem Fall zu beachten: Ein Startwert $x0$ wird benötigt! Für ihn darf natürlich nicht gelten, daß $f(x0)=0$ ist. Außerdem ist der Startwert so zu wählen, daß er nahe einer Nullstelle von f liegt. "Nah" ist dabei relativ. Das Bild 1 zeigt eine vernünftige Wahl von $x0$.

Bei Umsetzung in ein Programm ist unbedingt eine Begrenzung der Iteration einzuführen. Ist beispielsweise nach 50 Iterationen noch immer nicht eine vernünftig gewählte Genauigkeit erreicht, kann man davon ausgehen, daß das Verfahren divergiert, d. h. nicht konvergiert.

Unter geeigneten Umständen (f dreimal stetig differenzierbar, f hat an der gesuchten Stelle nur eine einfache Nullstelle, ...) konvergiert das Newton-Verfahren quadratisch, d. h. man nähert sich der Nullstelle bei der Iteration quadratisch. Sonst konvergiert das Newton-Verfahren nicht so gut.

Kompliziert aufgebaute Funktionen - etwa gebrochenrationale Funktionen in Sinus und Cosinus - erleichtern nicht gerade die Berechnung der ersten (oder der weiteren) Ableitungen. Das Newton-Verfahren nützt also nicht immer. Die Ableitung kann jedoch selbst auch genähert werden, beispielsweise durch eine passende Sekante. Der einzige Nachteil dabei ist, daß noch ein zusätzlicher Wert benötigt wird. Um ein solches Verfahren starten zu können, benötigt man zwei Startwerte.

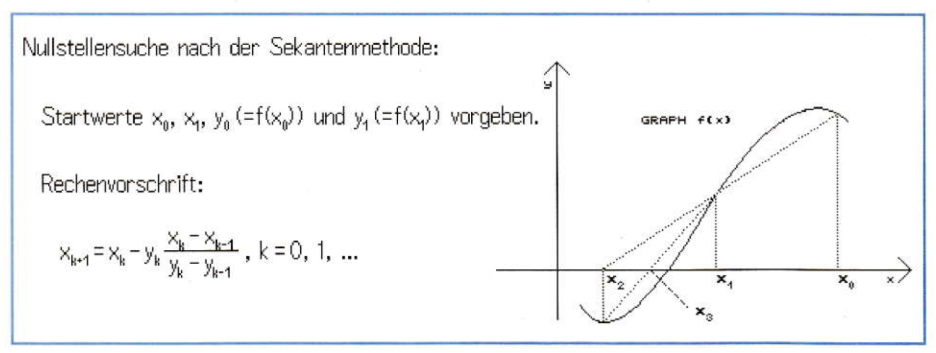


Bild 2

Die *resultierende Sekantenmethode* (vgl. Bild 2) ist dementsprechend ein zweistufiges Iterationsverfahren.

Sofern die erste und die zweite Ableitung an der gesuchten Nullstelle von *f* nicht selbst Nullstellen haben, konvergiert die Sekantenmethode mit einer Ordnung von ungefähr 1.618. Das Verfahren ist also schlechter als das Newton-Verfahren mit quadratischer Konvergenzordnung, was aber auch zu erwarten war, da zusätzlich die erste Ableitung der Funktion genähert werden muß. Das Beispielprogramm 2 zeigt, daß trotzdem kaum zusätzliche Iterationsschritte benötigt werden.

Ein einfaches Verfahren soll noch erwähnt werden. Es liegt im Trend der bisherigen beiden Verfahren: Die Konvergenzordnung ist nur linear, also noch schlechter als bei den beiden vorigen. Dafür ist das Verfahren sehr leicht zu implementieren, wie im Beispielprogramm 3 zu sehen ist. Man sucht sich ein Intervall *[a;b]*, so daß *f(a)* und *f(b)* unterschiedliches Vorzeichen bei wiederum stetigem *f* haben. Dann teilt man das Intervall und berechnet *f[(a+b)/2]*. War jetzt *f(a)* negativ bzw. positiv, und ist *f[(a+b)/2]* positiv bzw. negativ, wählt man als neue rechte Intervallgrenze einfach die Intervallmitte *(a+b)/2*. Analog wird die Intervallmitte als die linke Intervallgrenze genommen, falls *f(b)* und *f[(a+b)/2]* unterschiedliches Vorzeichen haben. Dann wird das Intervall wieder halbiert, und die Veränderung der Grenzen beginnt erneut. Es besteht immer die Sicherheit, daß eine gesuchte Nullstelle in dem betrachteten Intervall liegt, aber bis die gewünschte Genauigkeit erreicht ist, braucht man wesentlich mehr Schritte. Einen Vorteil hat das Verfahren trotzdem: Die Funktion *f* muß nur stetig sein.

Der Ursprung vieler Iterationsverfahren - auch des Newton-Verfahrens - ist die *Fixpunktiteration*. Für einen Fixpunkt *x* einer Funktion *F(x) gilt: F(x)=x*. Hat man einen Fixpunkt, so auch eine Nullstelle der Gleichung *F(x)-x=0*. Die Fixpunktiteration wurde im Beispielprogramm 4 auf eine mehrdimensionale Funktion angewandt und ermittelt eine nichttriviale Nullstelle. Für die mathematisch sehr Interessierten unter den Lesern möchte ich noch die Voraussetzungen nennen, die für die Konvergenz nötig sind. Es ist nach dem bekannten Banachschen Fixpunktsatz nötig, daß die Funktion *F* kontrahierend ist. Der Satz wurde nach Stefan Banach (geb. 30. 3. 1892 in Kraków, gest. 31. 8. 1945 in Lwów) benannt.

Das Newton-Verfahren ist durchaus auch bei mehrdimensionalen Funktionen anwendbar. Es wird jedoch dann statt der eindimensionalen Ableitung die Funktionalmatrix, d. h. die Matrix, in der alle Ableitungen stehen, oder noch genauer, die Inverse davon benötigt. Da dies meist nur noch mit numerischen Methoden zu bewältigen ist, braucht man ein Verfahren zur Näherung von Ableitungen (siehe Folge 3) und eines, mit dem man eine Matrix invertieren kann (siehe Folge 1). Aber das soll hier nicht mehr ausgeführt werden.

Fixpunktiteration:

Gesucht ist ein Fixpunkt von

$$F(x,y,z) := \begin{pmatrix} yz/6 \\ 5x-3 \\ xyz/2 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Bild 3

Manchmal möchte man statt der Nullstellen einer Funktion auch die Polstellen ermitteln, beispielsweise für eine Kurvendiskussion. Das geht auch mit den genannten Verfahren! Eine Polstelle liegt genau dann vor, wenn der Kehrwert der Funktion eine Nullstelle hat! Nehmen wir an, wir suchen eine Polstelle der Funktion *f(x)*. Wir definieren uns eine Funktion *g(x):=1/f(x)* und suchen dann die Nullstellen von *g*.

Ebenso gibt es elegante Hilfsmittel bei der Bestimmung von Nullstellen bei Polynomen. Ist eine Nullstelle *x0* bestimmt, kann man die Funktion durch *(x-x0)* dividieren, um dann Nullstellenbestimmungen für die so erhaltene Funktion durchzuführen. Hier tritt das bekannte Horner-Schema in Aktion. Aber auch dieses soll nicht mehr ausgeführt werden.

Insgesamt gibt es sehr, sehr viele numerische Algorithmen. Jeder erfüllt seinen Zweck, aber leider gibt es keinen, der als "Allzweck-Algorithmus" dienen könnte. Die Numerik ist aus unserem hochtechnisierten Leben gar nicht mehr wegzudenken. Schauen wir doch nur einmal mit einem engen Blick auf den ATARI ST. Ein Malprogramm, das Kurven berechnet, braucht einen numerischen Algorithmus.

Ergebnisse:

tatsächlicher Wert: 1.414213562

Intervallhalbierung:
Benötigte Schritte: 17
f(1.4142140151)=0

Sekanten-Methode:
Benötigte Schritte: 6
f(1.4142135382)=0

Newton-Verfahren:
Benötigte Schritte: 5
f(1.4142135623)=0

Fixpunktiteration:
Iterationsschritte: 11
Fixpunkt:
0.0
-3.0
0.0

Oder blicken wir weiter hinaus. Mondflug und Raketenstarts ohne numerische Algorithmen, die schnell und ausreichend genau die Flugbahn berechnen, wären nicht denkbar. Selbst Umweltschutz braucht die Numerik. Wie wollte man sonst die Ausbreitung von Schadstoffen bei Unfällen vorausberechnen? Beispiele für die Anwendung der numerischen Methoden in einem heutzutage doch populärem Bereich sind in [9] dargestellt.

Tja, da hätten wir das Ende der Artikelserie erreicht. Hat Sie die Angabe einiger weniger personenbezogener Daten gestört? Sicherlich nicht. Aber ist Ihnen mal etwas aufgefallen? Die Grundlage zu vielen heute benutzten Algorithmen stammt von Leuten, die keine Computer

TOS



Anlaß zu dieser kleinen Untersuchung war der Versuch, ein neues Programm [1] unter verschiedenen Umgebungen auszutesten. Da dieses Programm mit einigen neuen und interessanten Eigenschaften ausgestattet war und unter anderem den Aufruf der erst ab AES V1.3 implementierten Funktion *fsel_exinput* - gekoppelt an eine Versionsabfrage - enthielt, begaben wir uns also zu einem Bekannten, der eine Entwicklerversion des TOS 1.4 installiert hatte, das ja AES V1.3 enthalten sollte.

Doch obwohl wir alle ganz sicher waren, TOS 1.4 wirklich vor uns zu haben, griff die Abfrage offensichtlich nicht, und es war leider nichts mit dem erwarteten *fsel_exinput*!

Um diesem merkwürdigen Effekt auf die Spur zu kommen, setzte ich mich hin und schrieb ein kleines Assembler-Programm, das mir alle Versionsnummern und -daten der wichtigen Betriebssystemteile ausgeben würde. Die Überlegungen hierzu sowie zu den von ATARI bzw. Digital Research benutzten Formaten zusammen mit den Ergebnissen der Untersuchungen verschiedener mir bekannter TOS-Versionen möchte ich den geneigten 'Atarianern' nicht vorenthalten. Außerdem schienen mir nun, nachdem der ATARI ST inzwischen in ein - für Computer - schon 'reifes Alter' gekommen ist, Reflektionen zur Geschichte und Entwicklung seines Betriebssystems angebracht.

Das TOS

Das Betriebssystem des ATARI ST wurde ursprünglich in den USA bei Digital Research Inc. (DRI), den Schöpfern von CP/M und GEM, entwickelt und besteht aus mehreren Teilen, die aufeinander aufbauen: 'TOS' (The Operating System [2]), bestehend aus BIOS (Basic Input Output System), XBIOS (Extended BIOS) und GEMDOS (GEM Disk Operating System), stellen das 'eigentliche' Betriebssystem dar [2;7]. Darauf aufgesetzt ist GEM (Graphic Environment Manager), das wiederum aus VDI (Virtual Device Interface) und AES (Application Environment Services) besteht und beim ATARI ST, im Gegensatz zum PC-GEM, ein integraler Teil des Betriebssystems im weiteren Sinne ist - übrigens einer der Gründe, weshalb das ST-GEM vom Rechtsstreit zwischen Apple und DRI unberührt blieb. Dieser Auffassung entsprechen auch die Einteilung der auf dem ST lauffähigen Programme nach ihrer jeweiligen Oberfläche in TOS-Anwendungen und GEM-Applikationen, die Namensgebung der entsprechenden Extender und schließlich die Aufmachung der Copyright-Box im Desktop. Oft wird aber auch das ST-Betriebssystem als Ganzes unter dem Namen TOS zusammengefaßt, z.B. in [3]. Wie dem auch sei, das TOS trägt eine Versionsnummer und ein Entstehungsdatum, von den Teilen des Betriebssystems sind außerdem GEMDOS und AES mit gesonderten Versionsnummern versehen, die sich mit Hilfe spezieller Funktionen abfragen lassen. Dazu später mehr. Doch kommen wir nun zu dem Teil, in dem die meisten interessanten Informationen zu diesem Betriebssystem abgelegt sind, und das ist:

Der 'TOS-Header'...

... oder TOS-Programmkopf, denn das TOS ist ja schließlich nichts anderes als ein Programm, und zwar das Programm, das gestartet wird, wenn wir den ST einschalten, und das dann die ganze Zeit läuft - im Vordergrund oder, wenn andere Programme oder 'Applikationen' laufen, im Hintergrund. Wie kommen wir nun an diese interessanten Informationen heran?

Das TOS befindet sich bei den meisten STs in einem 192 kByte großen ROM (Read Only Memory), bis auf einige frühe Versionen dieses Rechners (260 ST, 520 ST und 520 ST+), die stattdessen nur ein sogenanntes Boot-ROM haben - eine 16 kByte große Miniausgabe, die den ST gerade einmal dazu befähigt, das 'richtige', d.h. vollständige TOS von Diskette nachzuladen (Näheres dazu und zu den Daten des Boot-ROMs weiter unten). Das ROM beginnt beim ST an der Adresse $FC0000.

Doch nun nur ja nicht etwa frech ins ROM 'gepeekt'! Denn schließlich muß das 'aktive' TOS ja gar nicht dort liegen. Es kann ja auch eine andere Version des TOS von Diskette nachgeladen worden sein, oder wir haben es gar mit einem der neuen STEs zu tun, bei denen das ansonsten kompatible TOS an der Adresse $E00000 liegt! Und wer weiß, was die Zukunft noch alles bringen wird. Also holt man sich die Adresse des TOS-Headers korrekterweise aus der garantierten Betriebssystemvariablen *_sysbase* ($4F2).

Dort liegen nun die uns hier interessierenden Daten an den Offsets, die man Tabelle 1 entnehmen kann (die Bezeichnungen wurden aus [3] übernommen):

Die drei Variablen *os_base*, *os_start* und *os_membot* habe ich eigentlich nur aufge-

Offset	Format	Bezeichnung	Bedeutung
$02	word	os_version	TOS-Versionsnummer
$04	long	os_start	TOS-Startadresse
$08	long	os_base	TOS-Header-Adresse
$0C	long	os_membot	Anfangsadresse des freien Speichers
$18	long	os_gendat	TOS-Erstellungsdatum im BCD-Format
$1E	word	os_gendatg	das gleiche im GEMDOS-Format, erst seit dem TOS vom 20.11.1985

Tabelle 1: Offsets aus der Betriebssystemvariablen _sysbase

nommen, um einmal zu verfolgen, wie sich die Lage des Betriebssystems im Laufe der Zeit und der fortschreitenden Versionsnummern immer weiter zu höheren Adressen hin verschoben hat, und wie auch der TOS-Header - schon zweimal - vergrößert, der OS-Pool [7] dann jedoch wieder verkleinert wurde.

Die TOS-Version

Da ist sie nun endlich, die TOS-Versionsnummer, verschwenderisch kodiert in einem Wort, jedes Byte eine Dezimalstelle enthaltend, und zwar so, wie ein M68000-Prozessor so etwas im Speicher abzulegen pflegt! Im höherwertigen Byte steht nämlich die Vorkomma- und im niederwertigen die Nachkommastelle. Man könnte dies sonst etwas ungewöhnliche Format als *decimal byte fixed*, also etwa 'Dezimal-Byte in Festkommadarstellung' bezeichnen. Das Wort $0102 z.B. ist also als Version 1.2 zu interpretieren, bekannter vielleicht unter dem Namen 'Blitter-TOS'. Der Vorteil ist hierbei, daß man es sehr einfach in ASCII wandeln kann, und im Hexdump ist es eben auch gut zu erkennen. Was die 'Verschwendung' betrifft so sind - oder waren - so viele TOS-Versionen wahrscheinlich auch gar nicht geplant. Hoffentlich reicht's beim jetzigen Entwicklungstempo bis zum Jahre 2099! Und damit kommen wir auch schon zum...

...TOS-Datum...

oder genauer TOS-Erstellungsdatum. Dies liegt im TOS-Header gleich zweimal - in verschiedenen Formaten - vor. Erst einmal als Langwort im Format *BCD (Binary Coded Decimal) fixed*, also etwa BCD-Festkommaformat. Die Kodierung ist MM/DD/YYYY, im angelsächsischen Format. $04221987 wäre also zu interpretieren als 04/22/1987 oder in unserer Schreibweise als 22.4.1987, das ist wieder das Datum des allseits bekannten Blitter-TOS'. Die Umwandlung des BCD-Formats, bei dem jedes Nibble (Halbbyte) eine Dezimalziffer enthält, ist relativ einfach zu bewerkstelligen, und außerdem läßt sich die Information direkt aus dem Hexdump ablesen.

Das gleiche Datum ist noch einmal am Offset $1E als Wort in GEMDOS- Kodierung abgelegt, allerdings - soweit mir bekannt - erst ab der TOS-Version vom 20.11.1985, als der TOS-Header zum erstenmal erweitert wurde, vorher begann hier schon der Code des Betriebssystems. Diese sog. GEMDOS-Kodierung ist die gleiche, die von den beiden GEMDOS-Funktionen *Tsetdate* und *Tgetdate* verwendet wird. Hier ist das Datum äußerst sparsam binär kodiert in den 16 Bits untergebracht, und zwar in folgender Form [4;5]:

Bit 0...4	Tag, von 1...31
Bit 5...8	Monat, von 1...12
Bit 9..15	Jahr seit 1980, von 0...119, d.h. bis 2099

Das schon erwähnte Datum des 'Blitter-TOS' erscheint hier in der Form $0E96, die Dekodierung ist also durch Bit-Schieben und Maskieren zu besorgen. Warum braucht man nun zweimal das gleiche Datum in verschiedenen Formaten? Ein Grund für diese Redundanz könnte gewesen sein, daß man sich eine vorher vorhandene Umwandlungsroutine ersparen wollte, um so das Ganze zu kürzen und schließlich in den 192 kByte ROM unterbringen zu können. Das war nämlich am Anfang ein ziemliches Problem für die TOS Entwickler.

TOS im RAM

Beim ST liegt das TOS ja, wie bereits erwähnt, an der Adresse $FC0000 im ROM, bis auf die Modelle eben, die hier nur ein sogenanntes 'Boot-ROM' haben. Diese müssen das TOS von Diskette nachladen, und es wird dann natürlich im RAM angelegt. Der Ablauf ist wie folgt: Das Boot-ROM initialisiert das System und sucht dann nach dem sog. *Boot-Loader* auf dem Boot-Sektor der System-Diskette. Dies ist ein kleines Programm, das die Systemdatei namens *TOS.IMG* an die Adresse $40000 (=256k) in den Speicher lädt. Hieraus kann man schon ersehen,daß die Mindestbestückung des ST mit RAM 512k betragen muß.(Ursprünglich geplante Auslegung in [5].) Dann wird die Startadresse von TOS.IMG (IMG = Image = Abbild) angesprungen. Hier befindet sich nun nicht etwa der TOS-Header, nein, liebe ST-Freunde, das wäre ja auch zu einfach (und zu unflexibel, dann würde man nämlich für jede Version einen anderen Boot-Lader brauchen!), es ist der sog. *Relocator (RELOCRL)* [8]. Dies ist wieder ein kleines Programm, das jetzt die Kontrolle übernimmt und das TOS.IMG (minus sich selbst) an seinen endgültigen Platz kopiert. Das TOS.IMG ist - wie die Kennung schon besagt - eine sog. 'Image-Datei', d.h. ein Abbild, das genau auf die absolute Adresse gelinkt wurde, an der es nachher im Speicher steht. Das hat den Vorteil, daß die darin enthaltenen absoluten Adressen nicht mehr reloziert werden müssen. Nun wird die Startadresse angesprungen, und das so aktivierte TOS erledigt den Rest der Initialisierung bis zum Erscheinen des beliebten GEM-Desktops oder einer Command-Shell.

Wie man sich leicht denken kann, unterscheiden sich die Abläufe während der Initialisierungsphase im RAM-TOS und ROM-TOS um einiges, deshalb ist es auch nicht möglich, sich ein TOS.IMG ins ROM zu brennen - von Copyright- Bedenken einmal abgesehen. Außerdem werden dort, wo das ROM-TOS Strukturen ins RAM kopiert, um sie da modifizieren zu können, beim RAM-TOS praktischerweise, und um Arbeitsspeicher zu sparen, diese Strukturen innerhalb des IMG selbst verändert. Das TOS.IMG gehört somit zur Gruppe der selbstmodifizierenden Programme, und so etwas ist im ROM nun einmal nicht zu verwirklichen! Ob RAM- oder ROM-TOS erzeugt werden soll, wird beim Systemhersteller ganz einfach durch das Linken unterschiedlicher Module erreicht, und die stehen ja nur den Systemprogrammierern der ATARI Corp. zur Verfügung.

Ein TOS.IMG von einem Speichermedium nachladen kann aber nicht nur das dazu bestimmte ursprüngliche Boot-ROM, diese Möglichkeit ist auch beim normalen ROM-TOS - gleich welcher Version - vorgesehen. Der Ablauf ist der gleiche, es braucht nur beim Start des ST eine sog. Systemdiskette mit Boot-Lader und TOS.IMG eingelegt zu sein. Nach dem Laden übernimmt dann das 'gebootete' TOS die Kontrolle über den ST. Das hat aber nicht etwa nostalgische Gründe! Wenn man noch Programme hat, die wegen 'spezieller' Progammierung auf dem aktuellen ROM-TOS nicht laufen

wollen, könnte man z.B. eine ältere Version des Betriebssystems als TOS.IMG booten. Oder man hat noch ein 'altes' TOS im ROM und möchte gern die Vorteile der jeweils neuesten Version genießen: Dann bootet man eben das neue TOS vom Massenspeicher. Denn es geht inzwischen nicht nur mit TOS.IMG und Boot-Lader von Diskette (langsam!), sondern auch mit Hilfe von Ladeprogrammen im Auto-Ordner z.B. von der Festplatte oder RAM-Disk (schnell bis sehr schnell!). Und es muß auch nicht immer ein TOS.IMG sein: Mir ist z.B. auch eine Version des 'Blitter-TOS', von dem m.E. nie ein .IMG im Umlauf war, bekannt, die mit Hilfe einer speziellen Relozierdatei (da hat sich jemand viel Arbeit gemacht!) und eines Laders im AUTO-Ordner gebootet wurde. Oder man denke an das KAOS [6], ein modifiziertes 'Blitter-TOS' mit einer auf MS-DOS gestylten Version des GEMDOS, das auf ähnliche Weise - allerdings an das zu diesem Zweck verschobene *phystop*, was leider die Möglichkeiten ziemlich einengt - geladen wird.

Auf jeden Fall, da es - wie auch sonst im Leben - nichts umsonst gibt, muß der 'Luxus' eines wie auch immer gebooteten TOS bezahlt werden, und zwar in bar mit ca. 200 kByte RAM.

GEMDOS

Das GEMDOS ist das DOS des TOS, oder, wie der Name (s.o.) schon sagt, eigentlich des GEM (Computersprach' - grauslich Sprach'!). Es trägt eine besondere, von der TOS-Version unabhängige Versionsnummer, die sich mit der Funktion *Sversion* erfragen läßt. Angeliefert wird von dieser Funktion ein Wort im Format *byte reversed, binary fixed* oder auch *Intel binary fixed*, kurzum ein binär kodierter Festkommawert mit vertauschten Bytes [4;5]. An diesem für M68k-Prozessoren 'haarsträubenden' Format kann man auch mal wieder sehen, was in den Köpfen der GEMDOS-Entwickler herumgeisterte, als sie das System entwarfen, und auch sonst ist ja sattsam bekannt [2;7], mit welch anderem System der Befehlssatz von GEMDOS die meisten Übereinstimmungen aufweist, und auf welcher Prozessorfamilie eben das läuft. Wundern wir uns also nicht weiter! Um wieder unser beliebtes Blitter-TOS als Beispiel anzuführen, der Wert $1300 ist also zu interpretieren als V 0.19, und diese Angabe bekommt man z.B. auch, wenn man im COMMAND.PRG (aus dem alten Entwicklungspaket) oder einer ähnlichen Text-Shell die Version des Betriebssystems abfragt. Daraus entnehme ich übrigens, daß die Kodierung der Versionsnummer binär und nicht etwa als BCD zu interpretieren ist, das ist nämlich in [4;5] nicht explizit angegeben. Die gleiche Versionsnummer bekam man übrigens auch schon beim TOS 1.0 vom 6.2.1986 geliefert, es war wohl alles mehr oder weniger beim alten geblieben. Die älteste GEMDOS-Versionsnummer, die ich zu sehen bekam, war - im Widerspruch zu [4] - V 0.13, aus dem deutschen 'Pilz-TOS' (so genannt, weil es statt der heutzutage üblichen Bomben Pilze ausgibt) vom 20.6.1985. Im TOS 1.4 vom 6.4.1989 ist - wie auch in den davor liegenden Testversionen - die GEMDOS-Version V 0.21 enthalten.

AES

Das oder die AES ist oder sind die oberste Schicht des Betriebssystems, auf der dann das GEM-Desktop als grafische Shell läuft. Als Teil des GEM von DRI geschaffen, trägt es (ich bleibe mal beim auch im Amerikanischen eingebürgerten Singular) auch wieder eine gesonderte und unabhängige Versionsnummer. Diese Nummer erhält man üblicherweise bei der Anmeldung einer Applikation - so heißen GEM-Programme - als ersten Eintrag im sog. *global*-Array, mit dem Namen *ap_version*.

Die Abfrage der AES-Version ist übrigens ein probates Mittel, um festzustellen, ob das AES vorhanden und initialisiert ist, z.B. bei Programmen, die sowohl auf der sog. TOS-Oberfläche als auch als GEM-Applikationen laufen sollen. Der Test auf *ap_id*, also die vom AES vergebene Applikationsnummer, bringt hier nämlich nichts, da sie für die Hauptapplikation üblicherweise Null ist. Dazu muß man wissen, daß das AES entweder noch nicht - während der Abarbeitung des AUTO-Ordners - oder auch nicht mehr - nach der Ausführung von *Puntaes* (XBIOS 39) bei RAM-TOS - installiert sein kann.

Das Format der Versionsnummer ist *BCD fixed*, also BCD-Festkommawert. Für die im 'Blitter-TOS' enthaltene Version des AES hat *ap_version* den Wert $0120, somit zu interpretieren als V 1.2. Auch im sog. 'alten' TOS vom 6.2.1986 hatte das AES schon diese Versionsnummer, was wohl heißt, daß sich da nichts oder zumindest nichts weiter Interessantes geändert hat.

Der Patch

Nachdem nunmehr die Voraussetzungen zu weiterer Erkundung geschaffen waren, begab ich mich wieder zu jenem Bekannten, um dort mein Programm auf seine Entwicklerversion des TOS 1.4 loszulassen. Es handelte sich übrigens um die Version vom 8.8.88 (schönes Datum!), die mit einer Alertbox *Launch TOS* ausgerüstet war. Das TOS hatte natürlich die Version 1.4, das GEMDOS die Version 0.21 und - man staune - das AES die Version 1.04! Also eine Rückentwicklung? Nein, das konnte nicht sein! Da mußte sich jemand geirrt und die Formate oder Versionsnummern verwechselt haben.

Wo war nun aber die AES-Versionsnummer im TOS abgelegt? Als Hilfe hatte ich erfahren, daß sie nicht etwa, wie von mir zuerst vermutet, in einer Tabelle steht, sondern vielmehr als *immediate* in einem Langwort zusammen mit *ap_count* in das *global*-Array geschrieben wird. Das bewußte Langwort $01040001 war leicht zu finden: Es kommt nämlich glücklicherweise nur ein einziges Mal im ganzen TOS vor! Der dazugehörige Befehl erwies sich als *move.l #$01040001,(a0)+* oder im Hexdump *$20FC01040001*. Nachdem nun dieser Wert vernünftig (z.B. in $01300001 oder $01400001) geändert und das System neu gebootet war, durften wir uns am eiwandfreien Arbeiten der Funktion *fsel_exinput* erfreuen.

AES-Versionsnummern

Die kleinste AES-Versionsnummer, die ich beobachten konnte, war 1.01 und stammt wieder aus besagtem alten 'Mushroom-TOS' vom 20.6.1985. Das sog. 'Beta-Test TOS' vom 18.5.1988 enthielt AES V1.3, am fortgeschrittensten sind das TOS 1.4 vom 22.2.1989, das erste sog. 'Rainbow-TOS' - nach den Farben, die wie ein Regenbogen das ATARI-Symbol durchlaufen - und dessen Nachfolger vom 6.4.1989, dort trägt das AES die Versionsnummer 1.4.

Vielleicht sollte man noch anmerken, daß es auch zur Zeit auf dem ATARI ST noch höhere AES-Versionsnummern geben kann, diese gehören dann aber nicht mehr zum TOS, sondern sind - gebootete - Aufsätze auf das TOS, wie z.B. die Portierung von GEM 2.0 der Firma ABC, in der das AES die Versionsnummer 2.1 oder neuerdings auch 2.2 trägt.

TOS-Versionsnummern

Die kleinste TOS-Versionsnummer hat natürlich das Boot-ROM, auch bekannt als 'Das Boot' [8], nämlich 0.0. Die ersten vollständigen TOS-Versionen bis einschließlich der bei uns wohl am weitesten verbreiteten (ROM und RAM) und oft als 'Altes TOS' bezeichneten Version vom 6.2.1986 tragen die Versionsnummer 1.0. Dann kommt das oben schon erwähnte 'Blitter-TOS' mit der Versionsnummer 1.2, interessanterweise im neuesten mir bekannten 'Diagnostic' von ATARI als 'OS Version #2' bezeichnet. Die neueste Ausgabe ist z.Z. das ebenfalls schon erwähnte 'Rainbow-TOS' V 1.4 vom 6.4.1989. Dies soll auch, wie man auf der ATARI-Messe in Düsseldorf vernehmen konnte, die 'zunächst' endgültige Version sein, die dann auch im Fachhandel auf ROMs zur Nachrüstung erhältlich sein wird.

Die Daten verschiedener TOS-Versionen, derer ich habhaft werden konnte, sind der besseren Übersicht halber noch einmal in Tabelle 2 zusammengefaßt.

TOS 1.6, ein an die veränderte und erweiterte Hardware des STE angepaßtes TOS 1.4, ist auf dem ST nicht lauffähig und ist daher in diesem Zusammenhang uninteressant. Allerdings sollten alle ST-Programme auch auf dem STE laufen können.

TOS	Datum	Name	Bytes	Sprache	Art	GEMDOS	AES	os_base	os_start	os_membot
0.0	-	Das Boot	16384	englisch	ROM	-	-	-	-	-
1.0	20.06.85	Mushroom	207128	deutsch	RAM	0.13	1.01	$5000	$501E	$19C00
1.0	20.11.85	-	197744	deutsch	RAM	0.19	1.2	$6100	$6120	$1A900
1.0	20.11.85	-	196526	franz.	RAM	0.19	1.2	$6100	$6120	$1A950
1.0	06.02.86	Altes	196480	deutsch	RAM	0.19	1.2	$6100	$6120	$1A950
1.0	06.02.86	Altes	196608	deutsch	ROM	0.19	1.2	$FC0000	$FC0020	$6100
1.2	22.04.87	Blitter	196608	deutsch	ROM	0.19	1.2	$FC0000	$FC0030	$8900
1.4	18.05 88	Beta-Test	195282	englisch	RAM	0.21	1.3	$AC00	$AC30	$2140C
1.4	08.08.88	Developer	196550	deutsch	RAM	0.21	1.04	$AD00	$AD30	$2140C
1.4	22.02.89	Rainbow	196608	deutsch	ROM	0.21	1.4	$FC0000	$FC0030	$611C
1.4	06.04.89	Rainbow	196608	deutsch	ROM	0.21	1.4	$FC0000	$FC0030	$611C
1.6	19.06.89	STE	196608	deutsch	ROM	0.21	1.4	$E00000	$E00030	$611C

Tabelle 2: TOS-Daten im Vergleich

Auf dem ATARI TT konnte man auf der Messe ja schon einige Programme laufen sehen. Das darauf installierte sog. TT-TOS soll, wie man bei ATARI in Düsseldorf vernahm, ein auf die Hardware des TT zugeschnittenes TOS 1.4 sein, aber der TT läßt ja noch auf sich warten. Welche Perspektive bietet sich nun uns Anwendern und Programmierern? ATARI hat angekündigt, das ST-Betriebssystem in Zukunft stärker als bisher mit Information und Dokumentation zu unterstützen - und das wurde zum Teil im Zusammenhang mit der Entwicklung der hier besprochenen neuen Versionen ja auch schon verwirklicht [9;10;11]. Warten wir's ab, was die Zukunft noch alles bringen wird! Auf jeden Fall wage ich die Prognose, daß dem ATARI ST, dieser vielseitigen Maschine mit ihrem faszinierenden Betriebssystem, noch viele Jahre interessanter Entwicklung bevorstehen.

Bernd Rosenlecher

Quellen:

[1] *Wolfram Roisch, METADUMP.APP, Programm und Assembler-Quelltext, 1989*

[2] *Tim Oren, Professional GEM, Column #15, Antic Publishing 1986*

[3] *Jankowski,Reschke,Rabich, ATARI ST Profibuch, Düsseldorf 1988*

[4] *Landon Dyer, ATARI GEMDOS Reference Manual, ATARI Corp. April, 1986. In den Erläuterungen zu 'Sversion' wird die Ausgabe vom 29.5.85 als 'first disk-based' und die vom 20.11.85 als 'first ROM-based' bezeichnet - es handelt sich hier natürlich um amerikanische Ausgaben - und beide sollen die GEMDOS-Versionsnummer $1300, d.i. V 0.19, tragen.*

[5] *The GEMDOS Programmer's Guide, Digital Research Inc. 1985*

[6] *Andreas Kromke, Das wahre GEMDOS, c't 11/88 S. 194 ff. Dem im KAOS V 1.2.3 enthaltenen GEMDOS wurde von seinem Autor die Versionsnummer V 0.21 erteilt.*

[7] *Kramer,Riebl,Hübner, Das TOS-Listing, Band 1, Hannover 1988*

[8] *Landon Dyer, A Hitchhiker's Guide to the BIOS, ATARI Corp. August, 1985*

[9] *TOS Release Notes, ATARI Corp., 1988, oft zitiert doch leider immer noch nicht allgemein zugänglich!*

[10] *RAINBOW TOS Release Notes, ATARI Corp., August, 1989, brandneu, s.o.!*

[11] *HMH (Hrsg.), SYSTEM_1.INF, US-Infos zum ATARI ST, Hamburg 1989. Bisher ca. 50 kByte Infos von ATARI und DRI Systemprogrammierern aus Usenet & anderen elektron. Quellen, auf Anfrage gegen Unkostenbeitrag auf Diskette erhältlich.*

Mein Dank gilt Herrn K.W. Quinckardt, Hamburg, der mir freundlicherweise seine reichhaltige Sammlung von originalen ATARI ST-Systemdisketten zur Auswertung zur Verfügung stellte.

```
 1: * ----------------------------------------------
 2: * test for aes initialization, TOS, GEMDOS & AES
     versions etc. (c) MAXON Computer GmbH 1989
 3: * ----------------------------------------------
 4: start:   bra.s     action
 5: * -------------------------------------------------
 6: * write word (max 5 decimal places) in d0 as
     decimal ASCII to string in a1
 7:
 8: dec_w:   move.w    #3,d1                ;4 dec. places
 9:          move.l    #10000,d2            ;4 dec. places
10: yeah:    divu      #10,d2
11:          ext.l     d0                   ;ready for div
12:          divu      d2,d0
13:          addi.b    #$30,d0              ;to ASCII
14:          move.b    d0,(a1)+             ;write
                                            to strg.
15:          swap      d0
16:          dbf       d1,yeah
17:
18:          rts
19: * -------------------------------------------------
```

→

```
20:  * kill leading or trailing zero in string
21:
22:  zero:     cmpi.b   #$30,-1(a1)          ;'0' ?
23:            bne.s    leave
24:
25:            move.b   #$20,-1(a1)          ;SPC
26:  leave:    rts
27:  * -----------------------------------------------
28:  * write long in d0 as hex to string in a1
29:
30:  wrthex: moveq   #7,d1        ;loop for 8 nibbles
31:  nibble: rol.l   #4,d0        ;get nibble
32:          move.l  d0,d2
33:          andi.b  #$0f,d2      ;mask
34:          addi.b  #48,d2       ;add '0' for '0'..'9'
35:          cmpi.b  #57,d2       ;>9?
36:          ble.s   wrt_it
37:
38:          addq.b  #7,d2        ;'A'..'F'
39:  wrt_it: move.b  d2,(a1)+     ;write to string
40:          dbf     d1,nibble
41:
42:          rts
43:  * -----------------------------------------------
44:  action:   move.l   4(sp),a0       ;basepage addr
45:            lea      mystk,a1       ;end of code
46:            move.l   a1,sp          ;new sp
47:            suba.l   a0,a1          ;prog length
48:
49:            move.l   a1,-(sp)             ;newsize
50:            move.l   a0,-(sp)             ;block
51:            clr      -(sp)                ;filler
52:            move.w   #$4A,-(sp)           ;Mshrink
53:            trap     #1                   ;GEMDOS
54:            lea      $C(sp),sp
55:
56:            lea      tos_dat(pc),a4 ;header msg
57:            bsr      print
58:
59:            pea      sysinfo(pc)
60:            move.w   #$26,-(sp)     ;Supexec
61:            trap     #14            ;XBIOS
62:            addq.l   #6,sp
63:
64:            movea.l  a0,a3          ;save sysbase
65:
66:            lea      os_base(pc),a4
67:            lea      18(a4),a1
68:            move.l   8(a3),d0             ;get
                                                os_base
69:            bsr      wrthex
70:            bsr      print                ;display
                                       addr of os_base
71:
72:            lea      os_start(pc),a4
73:            lea      18(a4),a1
74:            move.l   4(a3),d0             ;get
                                                os_start
75:            bsr      wrthex
76:            bsr      print                ;display
                                      addr of os_start
77:
78:            lea      os_memb(pc),a4
79:            lea      18(a4),a1
80:            move.l   $C(a3),d0            ;get os_
                                                membot
81:            bsr      wrthex
82:            bsr      print                ;display
                                     addr of os_membot
83:
84:            lea      tosver(pc),a4        ;message
                                                string
85:            lea      13(a4),a1            ;position
                                                to be
                                                patched
86:            move.b   2(a3),d0             ;get 1st
                                                byte
87:            addi.b   #$30,d0              ;to ASCII
88:            move.b   d0,(a1)              ;write
                                                it there
89:            addq     #2,a1                ;position
                                      to be patched
90:            move.b   3(a3),d0     ;get 2nd byte
91:            addi.b   #$30,d0      ;to ASCII
92:            move.b   d0,(a1)      ;write it there
```

```
93:          bsr       print           ;display
                                       TOS vers number
94:
95:          lea       bcd_date(pc),a4
96:          lea       16(a4),a1
97:          lea       $18(a3),a0
98:          move.b    (a0)+,d0               ;get
                                               month
99:          bsr       bcd_conv
100:         bsr       bcd_conv
101:         addq.l    #1,a1
102:         move.b    (a0)+,d0               ;get day
103:         bsr       bcd_conv
104:         bsr       bcd_conv
105:         addq.l    #1,a1
106:         move.b    (a0)+,d0               ;get 1st
                                     half of year
107:         bsr       bcd_conv
108:         bsr       bcd_conv
109:         move.b    (a0)+,d0               ;get 2nd
                                     half of year
110:         bsr       bcd_conv
111:         bsr       bcd_conv
112:         bsr       print                  ;display
                                     TOS BCD date
113:
114:         cmpi.b    #$1E,7(a3)             ;oldest
                                     TOS version ?
115:         beq.s     next
116:
117:         lea       dos_date(pc),a4
118:         lea       16(a4),a1              ;position
                                     to be patched
119:         move.w    $1E(a3),d3             ;get
                                     GEMDOS coded date
120:         move.w    d3,d0
121:         andi.w    #$1F,d0                ;get day
122:         bsr       decimal
123:         addq      #1,a1
124:         lsr       #5,d3                  ;get
                                     month to position
125:         move.b    d3,d0
126:         andi.b    #$F,d0                 ;get
                                               month
127:         bsr       decimal
128:         addq      #1,a1
129:         lsr       #4,d3                  ;get
                                     (year-1980) into
                                     position
130:         move.b    d3,d0
131:         andi.w    #$7F,d0                ;get
                                     (year-1980)
132:         addi.w    #1980,d0               ;year
133:         bsr       dec_w
134:         bsr       print                  ;display
                                     TOS DOS date
135:
136:
137: next:   move      #$30,-(sp)             ;Sversion
138:         trap      #1                     ;GEMDOS
139:         addq.l    #2,sp
140:
141:         tst       d0
142:         beq.s     close
143:
144:         move.w    d0,d7                  ;save
                                     version no
145:         lea       dosver(pc),a4          ;message
                                               string
146:         lea       12(a4),a1              ;position
                                     to be patched
147:         bsr       decimal
148:         rol       #8,d7
149:         move.b    d7,d0                  ;get
                                               highbyte
150:         addq      #1,a1                  ;advance
                                               position
151:         bsr       decimal
152:         bsr.s     print                  ;display
                                     GEMDOS version
153:
154:         moveq     #10,d0                 ;opcode
                                            appl_init
155:         move.l    #$00010000,d1          ;input
                                     to control array
156:         bsr.s     aes
```

→

```
157:            lea       global,a6
158:            move      4(a6),d4              ;ap_id
159:            bmi.s     close                 ;function
                                      failure->end prg
160:
161:            move.w    (a6),d5               ;ap
                                                 version
162:            beq.s     not_inst
163:
164:            lea       inst(pc),a4           ;message
                                                 string
165:            lea       12(a4),a1             ;position
                                      to be patched
166:            rol       #8,d5
167:            move.b    d5,d0                 ;get
                                                 highbyte
168:            bsr.s     bcd_conv              ;1st
                                                 nibble
169:            bsr       zero                  ;kill
                                                 zero
170:            bsr.s     bcd_conv              ;2nd
                                                 nibble
171:            addq      #1,a1                 ;advance
                                      position to be
                                      patched
172:            rol       #8,d5
173:            move.b    d5,d0                 ;get
                                                 lowbyte
174:            bsr.s     bcd_conv              ;1st
                                                 nibble
175:            bsr.s     bcd_conv              ;2nd
                                                 nibble
176:            bsr       zero                  ;kill
                                                 zero
177:  go_on:    bsr.s     print
178:            bsr.s     bconin
179:
180:            moveq     #19,d0                ;appl_
                                                 exit
181:            move.l    #$00010000,d1         ;input
                                      to control array
182:            bsr.s     aes
183:            move      (a6),d5               ;return
                                                 value
184:
185:  close:    clr -(sp)
186:            trap #1
187:  * -----------------------------------------------
188:  not_inst: lea       missing(pc),a4
189:            bra       go_on
190:  * -----------------------------------------------
191:  aes:      lea contrl,a0
192:            move d0,(a0)            ;opcode
193:            movep.l d1,3(a0)        ;fill
                                      parameter array
194:            move.l #aespb,d1        ;addr table
195:            move #$C8,d0            ;AES
196:            trap #2                 ;GEM
197:            rts
198:  * -----------------------------------------------
199:  * string output to console: modelled after
        Cconws, string addr in A4
200:
201:  print:    move.b    (a4)+,d0              ;string
                                                 addr
202:            beq.s     return
203:            move      d0,-(SP)              ;char
204:            move      #2,-(sp)              ;con
205:            move      #3,-(sp)              ;Bconout
206:            trap      #13                   ;BIOS
207:            addq.l    #6,sp
208:
209:            bra       print
210:  return:   rts
211:  * -----------------------------------------------
212:  bconin:   move      #2,-(sp)              ;con
213:            move      #2,-(sp)              ;bconin
214:            trap      #13
215:            addq.l    #4,sp
216:            rts
217:  * -----------------------------------------------
218:  * convert bcd-byte in d0 to 2 position decimal
        ascii - write to string in a1
219:
220:  bcd_conv: rol.b     #4,d0                 ;get
                                                 nibble
221:            move.b    d0,d1
222:            andi.b    #$F,d1                ;mask
223:            addi.b    #$30,d1               ;to ASCII
224:            move.b    d1,(a1)+              ;write
                                      char to string
225:            rts
226:  * -----------------------------------------------
227:  * convert byte in d0 to 2 position decimal ASCII
      - write to string in a1
228:
229:  decimal:  andi.l    #$FF,d0          ;byte only of
                                        interest here,
                                        .l for div!
230:            cmpi.b    #9,d0            ;single
231:            bls.s     single
232:
233:            divu      #10,d0           ;get tens
234:            bsr.s     chgform          ;output
235:            swap      d0               ;get units
236:  chgform:  add.b     #48,d0           ;to ASCII
237:            move.b    d0,(a1)+         ;write char to
                                        string
238:            rts
239:
240:  single:   move.b    #' ',(a1)+       ;leading zero
241:            bra       chgform          ;output
242:  * -----------------------------------------------
243:  sysinfo:  movea.l   $4F2,a0          ;sysbase
244:            rts
245:  * -----------------------------------------------
246:  aespb:    dc.l contrl,global,intin,intout,addrin,
                  addrout
247:
248:  * the following are fixed structs after each
      label, XX etc are to be patched
249:
250:  inst:     dc.b      13,10,' AES     V XX.XX
                          resident ',13,10,0
251:
252:  missing:  dc.b      13,10,' AES not (yet?)
                          resident ',0
253:
254:  dosver:   dc.b      13,10,' GEMDOS V XX.XX
                          resident ',0
255:
256:  tosver:   dc.b      13,10,' TOS     V  X.X
                          resident ',0
257:
258:  bcd_date: dc.b      13,10,' TOS BCD date MM/DD/
                          YYYY ',0
259:
260:  dos_date: dc.b      13,10,' TOS DOS date
                          DD.MM.YYYY ',0
261:
262:  os_base:  dc.b      13,10,' os_base
                          $00XXXXXX ',0
263:
264:  os_start: dc.b      13,10,' os_start
                          $00XXXXXX ',0
265:
266:  os_memb:  dc.b      13,10,' os_membot
                          $00XXXXXX ',10,0
267:
268:  tos_dat:  dc.b      27,'E'
269:            dc.b      13,10,' TOS VERSION INF
                          MAXOnN Computer 89 ',10,0
270:  * -----------------------------------------------
271:            bss
272:            even
273:  contrl:   ds    5
274:
275:  global:   ds   15
276:
277:  intin:    ds   16
278:
279:  intout:   ds    7
280:
281:  addrin:   ds.l  3
282:
283:  addrout:  ds.l  1
284:
285:            ds.l 50
286:  mystk:    ds.b  1
287:  * -----------------------------------------------
```

Patch As Patch Can

Modifizierte TOS 1.4-EPROMs im MEGA ST

Seit neuestem ist das TOS 1.4 (offizielle Version vom 6.4.1989) von ATARI auf dem Markt, in dem viele Fehler aus früheren TOS-Versionen behoben worden sind und das eine ganze Reihe von neuen Funktionen bietet (Beschreibung siehe ST 11/89). Aber wie nicht anders zu erwarten war, ist auch dieses neueste Betriebssystem für den ST nicht fehlerfrei.

Neben einigen sehr alten Fehlern - z.B. dem Boot-Device-Fehler, der bei allen Patch-Vorschlägen (Patch = direkte Korrektur in Programmen) erwähnt wird - haben sich auch einige neue Fehler eingeschlichen. Für ATARI-Besitzer, die ein Modem oder anderes Gerät an der seriellen Schnittstelle betreiben wollen, ist ein Fehler beim Handshake dieser Schnittstelle besonders ärgerlich.

Hat man sich also erst einmal das neue TOS besorgt, kann es durchaus lohnend sein, diese Fehler durch Patches zu beheben und das modifizierte Betriebssystem in EPROMs zu brennen. Außerdem kann man dabei gleich die Default-Einstellungen des Desktops und ähnliche Dinge seinem eigenen Geschmack anpassen.

Voraussetzung dafür ist in jedem Falle erst einmal, daß man sich sein TOS 1.4 auf Diskette abspeichert. Dafür finden Sie in Listing 1 das Programm *ROMSAVE*, das einfach eine Kopie des eingebauten Betriebssystems unter dem Namen *TOS.IMG* auf die Diskette speichert. Nun können Sie beliebige Änderungen vornehmen, was im ROM nur schwerlich zu bewerkstelligen wäre.

Beim Kauf der EPROMs hat man die Wahl zwischen sechs kleinen 256 kBit-Chips (32 k * 8 Bit, Bezeichnung 27C256) oder zwei großen 1 MBit-Chips (128 k * 8 Bit). Zwei 1 MBit-EPROMs haben eine Kapazität von 256 kByte, von denen allerdings im ST nur 192 kByte adressiert werden, während der Rest unbenutzt bleibt. So etwas wie 96 k * 8 Bit gibt es jedoch nicht, weil der Adreßraum von EPROMs immer eine 2er-Potenz ist.

Die Lösung mit sechs EPROMs hat vor allem den Vorteil, daß sie wesentlich billiger ist. Zwei 1 MBit-Typen kosten fast das Doppelte des Preises von sechs 256 kBit-Chips.

Um die 192 kByte lange Datei *TOS.IMG*, die *ROMSAVE* erzeugt hat, nach dem Patchen in die 256k-EPROMs zu brennen, muß sie vorher noch in die entsprechenden sechs Teile zerlegt werden. Dazu dient das Programm *SPLIT.TTP* aus Listing 2. In der Kommandozeile des Programms muß der Name der zu zerlegenden Datei, also z.B. *TOS.IMG*, übergeben werden. *SPLIT.TTP* erzeugt dann sechs Dateien mit demselben Namen und den Extensions *.LO0*, *.LO1*, *.LO2*, *.HI0*, *.HI1* und *.HI2*. Die Aufteilung des Betriebssystems auf die geraden und ungeraden Adressen der verschiedenen Adreßbereiche ist aus der Tabelle abzulesen. *SPLIT* und auch *ROMSAVE* sind in Laser C geschrieben worden, sollten sich aber nach ein paar Änderungen auch mit jedem anderen C-Compiler übersetzen lassen. Eventuell ist #include <osbind.h> in der ersten Zeile in #include <tos.h> umzuwandeln.

Der Umbau

Wenn man einen schon sehr betagten Rechner hat, wird es überhaupt keine Probleme beim Umbau geben. Man nimmt einfach die sechs alten ROMs aus der Fassung und ersetzt sie durch die neuen, frisch gebrannten EPROMs. Dabei sollte man darauf achten, daß man jeden Chip in die richtige Fassung steckt, da andernfalls auch das beste TOS keine Chance hat, dem Rechner irgendetwas zu entlocken.

Bezeichnung alt	neu	Adreßbereich	
U7	LO-0	$FC0000-$FCFFFF	ungerade Adressen
U6	LO-1	$FD0000-$FDFFFF	ungerade Adressen
U5	LO-2	$FE0000-$FEFFFF	ungerade Adressen
U4	HI-0	$FC0000-$FCFFFF	gerade Adressen
U3	HI-1	$FD0000-$FDFFFF	gerade Adressen
U2	HI-2	$FE0000-$FEFFFF	gerade Adressen

Tabelle 1: So ist das TOS auf sechs 256k-(EP)ROMs verteilt.

Etwas problematischer gestaltet sich dieser Umbau bei den neuen MEGA STs, da diese mit zwei großen ROMs geliefert werden, die man nicht ohne weiteres gegen sechs EPROMs auswechseln kann. ATARI hat zwar auf der Platine sechs Sockel für das TOS untergebracht, und auch die gesamte Schaltung zu ihrer Ansteuerung ist schon für diesen Fall vorbereitet, jedoch muß man der zuständigen Hardware diese Änderung irgendwie mitteilen.

In einer Umbauanleitung, deren Ursprung ich nicht kenne, heißt es, man müsse zwei (schon vorhandene) Widerstände umlöten, also zwischen zwei andere Lötlöcher

Bild 1: Änderungen am Platinenlayout, um sechs 256k- statt 2 1MB-Bit-(EP)ROMs verwenden zu können.

auf der Platine eine zusätzliche Lötbrücke W4 einlöten und außerdem Pin 12 des ICs U12 durchkneifen.

Diese Methode ist jedoch nicht nur sehr brutal - wenngleich das IC U12, ein 74LS11, auch nur 'n paar Pfennige kostet - sondern sie ist auch nur sehr schwer wieder rückgängig zu machen. Sie stehen also vor einem Problem, wenn Sie möglicherweise Ihr altes Betriebssystem nochmal benutzen wollen, weil z.B. ein wichtiges Programm nicht mit dem neuen TOS läuft.

Nach langem Suchen und Verfolgen der Leiterbahnen auf der Platine mit einem Durchgangsprüfer fand ich eine wesentlich bessere Lösung, die zwar auch nicht ohne Löten auskommt (das ist prinzipiell nicht möglich), die aber jederzeit ein einfaches und schnelles Umstellen von zwei auf sechs EPROMs bzw. umgekehrt ermöglicht - und das dann ohne zu löten.

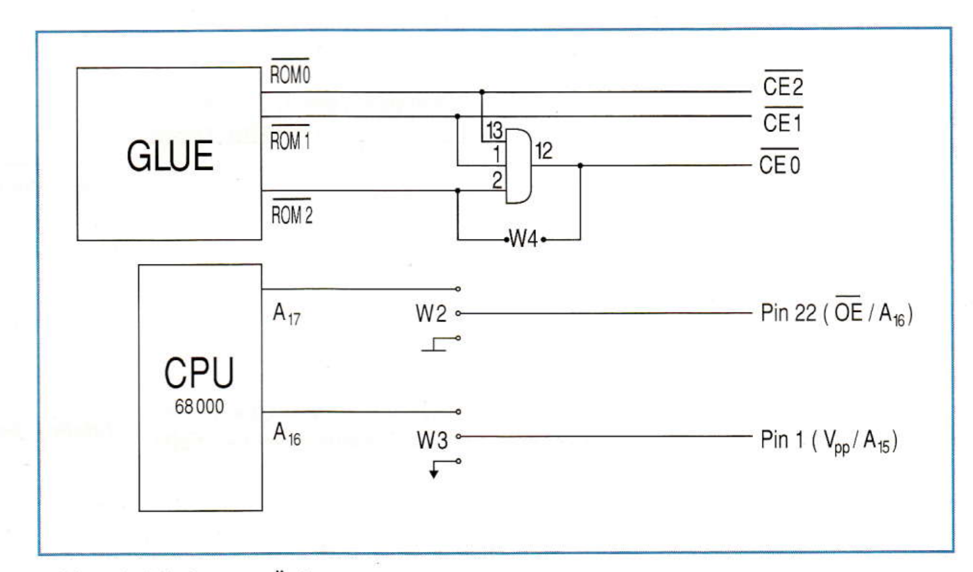


Bild 2: Die Schaltung zur Änderung

Zunächst fällt auf, daß die beiden umzulötenden "Widerstände" den Wert 0Ω (ja, tatsächlich Null) haben. Solche "Widerstände" werden hegestellt, weil die Bestückungsmaschinen damit leichter umgehen können als mit einfachen Drähten. Außerdem muß jeder der Widerstände zwischen zwei von drei Lötlöchern geschaltet werden (siehe Bild 1), und zwar zwischen das rechte und mittlere oder das mittlere und linke. Die Löcher haben einen Abstand von 2.54mm (1/10 inch), so daß man hier leicht sogenannte Jumper einbauen kann. Das sind drei senkrecht nebeneinanderstehende kleine Metallpfosten, von denen entweder die beiden rechten oder die beiden linken mit einem Steckkontakt verbunden werden. Auf der Platine sind diese beiden Widerstände mit den Namen W2 und W3 gekennzeichnet.

Das IC U12, ein 74LS11, enthält drei AND-Gates mit den jeweiligen Eingängen. Mit einem Durchgangsprüfer und etwas Geduld findet man heraus, daß die Eingänge zweier dieser Gates auf Masse gelegt, die Gates also unbenutzt sind, und ein Blick in das Datenblatt zeigt, daß Pin 12 der Ausgang des letzten Gates ist. Kneift man diesen durch, hat man also die gleiche Wirkung, als wenn das IC gar nicht da wäre. Es bietet sich daher an, das IC ganz auszulöten und durch einen Sok-

Patch 1

Beim Warmstart wurde bisher das GEMDOS-Datum auf das Erstellungsdatum des TOS (6.4.1989) gestellt und dann geprüft, ob die MEGA-Uhr vorhanden ist. Falls ja, wurde ihr Inhalt ins GEMDOS-Datum und die GEMDOS-Zeit übertragen. Jetzt wird zusätzlich, falls die MEGA-Uhr nicht vorhanden ist, die Uhr des Tastaturprozessors gelesen und - falls sie einen gültigen Wert hat - als GEMDOS-Zeit übernommen. ATARI hat im neuen TOS diese Funktion schon implementiert, durch einen Programmierfehler arbeitete sie jedoch nicht richtig. ST-Besitzer, die keine batteriegepufferte Uhr haben, brauchen die Zeit jetzt nur noch beim Einschalten des Rechners zu stellen.

Abfrage, ob MEGA-Uhr vorhanden

```
FC0448 6A18        bpl.s  $FC0462
```

Änderung in EPROM U4 / HI-0

Patch 2

Der Boot-Device-Fehler ist behoben, so daß der Environment-String richtig gesetzt wird.

_bootdev richtig auslesen

```
FC04B8 3039 ...    move.w  $000446.l,d0
```

Änderung in EPROM U4 / HI-0

Patch 3

Eine falsche Stack-Korrektur in der Routine, die Auto-Ordner-Programme ausführt, konnte unter ungünstigen Umständen zu Abstürzen führen.

Stackpointer korrigieren in autoexec

```
FC0C7C DEFC 000C  add.w   #12,a7
FC0C80 4A40       tst.w   d0
FC0C82 6668       bne.s   $FC0CEC
FC0C84 3F3C 0007  move.w  #7,-(a7)
FC0C88 2F38 0984  move.l  $0984,-(a7)
```

Änderungen in EPROM U4 / HI-0 und U2 / LO-0

Patch 4

Auch der bekannte Fastload-Patch ist hier angegeben, welcher aber mit Vorsicht zu genießen ist. Einige Laufwerke - besonders von NEC - neigen bei ihm zu wesentlich größerer Fehlerhäufigkeit. Ich habe aus diesem Grund diesen Patch nicht in mein EPROM eingebaut, zumal sich fast die gleiche Geschwindigkeit durch speziell formatierte Disketten erreichen läßt (Sektorversatz).

Fastload

```
FC1516 7C10        moveq  #$10,d6
```

Änderung in EPROM U7 / LO-0

Patch 5

Dieser Patch ist dem AUTO-Ordner-Programm von ATARI entnommen und behebt einen Fehler bei der Einstellung der seriellen Schnittstelle.

Rsconf (XBIOS 15)

```
FC3A3E  B07C 0003  cmp.w   #3,d0
FC3A42  6214       bhi.s   $FC3a58
FC3A44  660E       bne.s   $FC3A54
FC3A46  6008       bra.s   $FC3A50
FC3A48  4E71       nop
FC3A4A  4E71       nop
FC3A4C  4E71       nop
FC3A4E  4E71       nop
```

Änderungen in EPROM U4 / HI-0 und U2 / LO-0

Patch 6

Auch dieser Patch stammt aus ATARIs Patch-Programm und behebt einen Fehler des AES bei der Suche des Dateinamens in einer kompletten Pfadangabe.

AES: Dateinamen im Pfad finden (LINE-F: $F5DC)

```
FE411E  226F 0004  move.l  4(a7),a1
FE4122  2449       move.l  a1,a2
FE4124  4A19       tst.b   (a1)+
FE4126  66FC       bne.s   $FE4124
FE4128  1021       move.b  -(a1),d0
FE412A  B3CA       cmp.l   a2,a1
FE412C  650C       bcs.s   $FE413A
FE412E  B03C 005C  cmp.b   #$5C,d0
FE4132  6706       beq.s   $FE413A
FE4134  B03C 003A  cmp.b   #$3A,d0
FE4138  66EE       bne.s   $FE4128
FE413A  5289       addq.l  #1,a1
FE413C  2009       move.l  a1,d0
FE413E  4E75       rts
FE4140  4E71       nop
FE4142  4E71       nop
FE4144  4E71       nop
FE4146  4E71       nop
FE4148  4E71       nop
```

Änderungen in EPROM U2 / HI-2 und U5 / LO-2

kel zu ersetzen. Will man wieder mit zwei großen (EP)ROMs arbeiten, braucht man nur das IC wieder in den Sockel zu stekken. Beim Auslöten des ICs kann man sich eine Menge Arbeit sparen, indem man alle Pins mit einer Zange durchkneift und die Pins einzeln auslötet. Ein neues IC desselben Typs kann man in jedem Elektronikgeschäft kaufen. Bevor man den neuen Sockel einlötet, sollte man mit einer Entlötpumpe alles überflüssige Lötzinn absaugen.

Als letztes bleibt dann noch die Lötbrücke W4, die in der Nähe von IC U12 geschlossen werden muß. Auch hier waren wieder ein Durchgangsprüfer und noch mehr Geduld erforderlich, um sich das Ein- und Auslöten beim Umstellen zu sparen. Die

Patch	Adresse	alt	neu
(1)	$448	$6418	$6A18
(2)	$4B8	$1039	$3039
(3)	$C7C	$5C4F	$DEFC
		$4A40	$000C
		$666A	$4A40
		$3F3C	$6668
		$0007	$3F3C
		$2F39	$0007
		$0000	$2F38
(4)	$1516	$7C14	$7C10
(5)	$3A40	$FFFF	$0003
		$6714	$6214
		$1140	$660E
		$0020	$6008
		$670A	$4E71
		$C03C	$4E71
		$00FD	$4E71
		$6704	$4E71
(6)	$2411E	$4E56	$226F
		$0000	$0004
		$48E7	$2449
		$0104	$4A19
		$2A6E	$66FC
		$0008	$1021
		$4A1D	$B3CA
		$66FC	$650C
		$6002	$B03C
		$538D	$005C
		$BBEE	$6706
		$0008	$B03C
		$650C	$003A
		$0C15	$66EE
		$005C	$5289
		$6706	$2009
		$0C15	$4E75
		$003A	$4E71
		$66EC	$4E71
		$528D	$4E71
		$200D	$4E71
		$F801	$4E71

Tabelle 2: Nach Methode 1 wird die Datei TOS.IMG mittels eines Diskmonitors verändert und dann erst aufgesplittet.

beiden Lötlöcher, die mit einem Draht verbunden werden sollen, sind auf der Platine mit Pin 12 bzw. Pin 2 von U12 verbunden. Da die Lötbrücke aber nur dann gesetzt werden muß, wenn man mit sechs EPROMs arbeitet, wenn also das IC U12 nicht in seiner Fassung steckt, kann man sich einen Draht zurechtbiegen, der genau zwischen diese beiden Pins des Sockels paßt und statt des ICs eingesteckt wird.

Hat man diese Arbeiten erledigt, also die "Widerstände" durch Jumper und das IC durch einen 14poligen Sockel ersetzt, kann man seinen Lötkolben in der "Bastelkiste" liegenlassen, wenn man seinen Rechner umstellen will, da nur noch gesteckt werden muß.

Patch	EPROM	Adresse	alt	neu
(1)	HI-0	224	6461	6A61
(2)	HI-0	25C	1000	3000
(3)	HI-0	63E	5C4A	DE00
			663F	4A66
			002F	3F00
			0009	2F09
	LO-0	63E	4F40	FC0C
			6A3C	4068
			0739	3C07
			0084	3884
(4)	LO-0	A8B	1424	1024
(5)	HI-0	1D20	FF67	0062
			1100	6660
			67C0	4E4E
			0067	4E4E
	LO-0	1D20	FF14	0314
			4020	0E08
			0A3C	7171
			FD04	7171
(6)	HI-2	208F	4E00	2200
			4801	244A
			2A00	6610
			4A66	B365
			6053	B000
			BB00	67B0
			650C	0066
			0067	5220
			0C00	4E4E
			6652	4E4E
			20F8	4E4E
	LO-2	208F	5600	6F04
			E704	4919
			6E08	FC21
			1DFC	CA0C
			028D	3C5C
			EE08	063C
			0C15	3AEE
			5C06	8909
			153A	7571
			EC8D	7171
			0D01	7171

Tabelle 3: Nach Methode 2 werden die Änderungen erst nach dem Aufsplitten des TOS in sechs Teile vorgenommen.

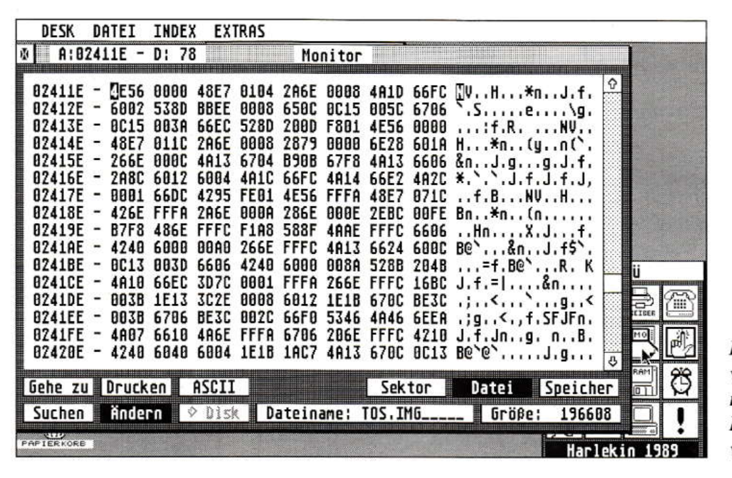


Bild 3: Bei Methode 1 wurden die Patches mit einem Diskmonitor vorgenommen.

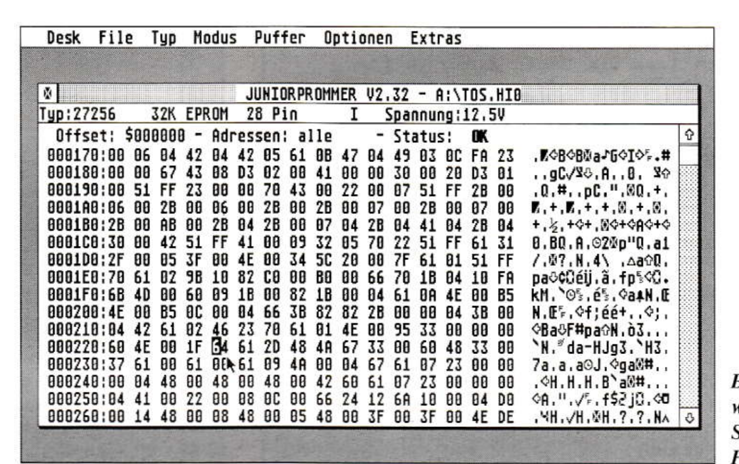


Bild 4: Bei Methode 2 wurde direkt die Software des Junior Prommers benutzt.

Und was haben wir jetzt angerichtet?

Für diejenigen, die auch daran interessiert sind, was sie eigentlich machen, wenn sie diesen Umbau vornehmen, folgt noch eine kurze Beschreibung der Schaltung zur Ansteuerung der EPROM-Sockel. Diese Schaltung ist in Bild 2 zu sehen.

In dem Zustand, in dem der Rechner ausgeliefert wird, ist das AND-Gate eingesetzt und die Lötbrücke W4 offen. Wird jetzt auf das ROM zugegriffen, geht je nach Adreßbereich (FC0000-FCFFFF, FD0000-FDFFFF oder FE0000-FEFFFF) eine der Leitungen /ROM2, /ROM1 oder /ROM0 vom GLUE auf low. Über das AND wird das Signal /CE0 low, wodurch die EPROMs in den ersten beiden Sockeln selektiert werden. In den beiden ROM-Adreßbereichen außer dem ersten werden zusätzlich /CE1 oder /CE2 aktiviert (low). Dies spielt aber keine Rolle, da in den entsprechenden Sockeln ja sowieso keine EPROMs stecken.

Fehlt das AND-Gate, und wird dafür die Lötbrücke W4 geschlossen, werden die Leitungen /ROM2, /ROM1 und /ROM0 eins zu eins auf die Leitungen /CE0 bis / CE2 durchgeschleift. Über die beiden Jumper W2 und W3 läßt sich einstellen, welches Signal auf den Pins 22 bzw. 1 der sechs EPROM-Fassungen anliegen. Steckt man beide Jumper nach links (256 kBit-Chips), so liegt Pin 22, also der Output-Enable-Eingang (/OE) des 27C256, auf Masse und Pin 1, der V_{PP}-Anschluß, auf +5V. Der 1 MBit-Chip hat wegen seines größeren Adreßraums zwei Adreßleitungen mehr, nämlich A_{15} auf Pin1 und A_{16} auf Pin 22. Bei 1 MBit-EPROMs müssen beide Jumper rechts stehen, wodurch diese beiden Pins mit A_{16} und A_{17} der CPU verbunden werden. Die Nummern der Adreßleitungen der CPU sind jeweils um eins größer als die des EPROMs, weil die CPU MC68000 aufgrund ihres 16 Bit-Datenbusses keine Adreßleitung A_0 besitzt.

TOS-Patches

Da auch den ATARI-Entwicklern Fehler unterlaufen, wird mit TOS 1.4 eine Diskette mit diversen Zusatzprogrammen ausgeliefert. Eins davon, das *TOS14FIX*-Programm, ist für den Auto-Ordner bestimmt und korrigiert einige fehlerhafte Betriebssystemroutinen. Um es überflüssig zu machen und auch noch nicht korrigierte Fehler zu beheben, haben wir Ihnen sechs Patches zusammengestellt, die Sie leicht mittels eines EPROM-Brenners

```
 1:  /* ROMSAVE.C
 2:     utility for copying the TOS-ROM to file
        TOS.IMG
 3:     by Urs Thürmann/SH    (c) MAXON Computer 89
 4:  */
 5:
 6:  #include <osbind.h>
 7:
 8:  #define ROM (char *)0xFC0000L
 9:  #define SIZE (long)(192 * 1024)
10:
11:  int main()
12:  {
13:     int fh;
14:     long index, stack;
15:     char *adr=ROM, *buf, *buf_start;
16:
17:     stack = Super(0L);
18:
19:     buf= buf_start=(char*)Malloc(SIZE);
20:     if (buf< (char*)-1)
21:        return(-1);
22:     Cconws("Ich lese die Daten ein...\n");
23:     for (index=0; index<SIZE; index++)
24:        *buf++ = *adr++;
25:
26:     Super(stack);
27:
28:     if ((fh = Fcreate("tos.img",0)) <0)
29:     {
30:        Cconws("Datei konnte nicht erzeugt
                   werden\r\n");
31:        return(-2);
32:     }
33:     Cconws("Ich schreibe die Daten auf
              Disk...\n");
34:     Fwrite(fh, SIZE,buf_start);
35:     Fclose(fh);
36:
37:     Mfree(buf_start);
38:     return(0);
39:  }
40:
```

Listing 1: Das ROMSAVE-Programm erstellt die Datei TOS.IMG.

```
 1:  /* SPLIT.C
 2:     utility for splitting a 192K TOS-file into 6
 3:     files by Urs Thürmann/SH (c) MAXON Computer 89
 4:  */
 5:
 6:  #include <osbind.h>
 7:  #include <string.h>
 8:
 9:  #define SIZE 192 * 1024L
10:
11:  char infile[SIZE], outfile[SIZE / 6];
12:
13:  int main(argc, argv)
14:  int argc;
15:  char *argv[];
16:
17:  {  char name[128], *ext, *strrchr();
18:     char *sptr, *dptr;
19:     int fh, i;
20:
21:     if(argc > 1)
22:        strcpy(name, argv[1]);
23:     else
24:        strcpy(name, "tos.img");
25:     if((fh = Fopen(name, 0)) < 0)
26:        return(-1);
27:     Fread(fh, SIZE, infile);
28:     Fclose(fh);
29:
30:     if(!(ext = strrchr(name, '.')))
31:        ext = strrchr(name, 0);
32:     for(i = 0; i < 6; i++)
33:     {
34:        for(dptr = outfile, sptr = infile + i/2 *
              (SIZE / 3) + i%2;
35:           dptr < outfile + SIZE / 6; dptr++,
              sptr += 2)
36:           {
37:              *dptr = *sptr;
38:           }
39:        if(i % 2)
40:           strcpy(ext, ".lox");
41:        else
42:           strcpy(ext, ".hix");
43:        ext[3] = i / 2 + '0';
44:        printf("Schreibe die Datei %s...\n",name);
45:        fh = Fcreate(name, 0);
46:        Fwrite(fh, SIZE / 6, outfile);
47:        Fclose(fh);
48:     }
49:     return(0);
50:  }
51:
52:  char *strrchr(text, ch)
53:  char *text;
54:  int ch;
55:  {
56:     int i = strlen(text);
57:
58:     while(i>=0)
59:     {
60:           if (text[i] == ch)
61:           return(&text[i]);
62:        i--;
63:     }  return(0);
64:  }
```

Listing 2: Das SPLIT-Programm teilt TOS.IMG auf sechs EPROMs auf.

und der beiden abgedruckten Programme in Ihrem TOS 1.4 unterbringen können. Natürlich braucht man nicht alle Patches vorzunehmen, denn jeder ist unabhängig von den anderen. Dabei gibt es zwei Vorgehensweisen:

1. Man nimmt die Änderungen mittels eines Disk-Monitors (s. Bild 3) direkt an der mittels *ROMSAVE* gespeicherten Datei *TOS.IMG* vor, was der einfachste Weg ist. Dazu benötigen Sie Tabelle 2. Die Änderungsadressen gehen von einer Offset-Adresse von $0 aus ($FC0000 entspricht $0), da Ihr Disk-Monitor mit Sicherheit genauso funktioniert. Nach den Änderungen speichern Sie einfach die ganze Datei wieder ab und starten dann das *SPLIT*-Programm, geben *TOS.IMG* ein und erhalten sechs Dateien, die Sie dann nur noch anhand Ihres Handbuches in Ihre EPROMs zu brennen brauchen.

2. Man speichert das TOS mittes *ROMSAVE* ab, splittet die Datei *TOS.IMG* mittels *SPLIT.TTP* auf und kann dann jede der sechs EPROM-Dateien einzeln bearbeiten. Hier benötigt man keinen Disk-Monitor, da alle Änderungen direkt mit der Software eines EPROM-Brenners (s. Bild 4, in unserem Beispiel der MAXON-Junior Prommer) vorgenommen werden können. Um es Ihnen auch hier recht einfach zu machen, haben wir Ihnen die zu ändernden Daten bereits aufbereitet. Aufbereitet deswegen, weil je drei EPROMs die geraden und drei die ungeraden Adressen des TOS enthalten müssen. In Tabelle 3 sind die EPROM-Bezeichnung, die Änderungsadresse und die bereits nach geraden und ungeraden Adressen aufgesplitteten Daten enthalten. Auch hier wird wie bei 1. wieder von einer Offset-Adresse von $0 ausgegangen. Am einfachsten benutzen Sie die Funktion *Gehe Adresse...*, mit der Sie jede Adresse direkt anspringen können. Jede Datei kann direkt nach der Veränderung gebrannt werden.

Achten Sie darauf, daß Sie die EPROMs nicht vertauschen und nach dem Brennen richtig beschriften, sonst geht nichts!

Urs Thürmann / HE

XBRA

Vektorverbiegende Programme

In der ST-Computer wurde schon mehrfach das XBRA-Verfahren für "vektorverbiegende" Programme vorgestellt. Solche Programme ändern die Adressen von Unterprogrammen in bestimmten Systemvariablen des ST. Auf diese Art lassen sich etwa Treiber für RAM-, ROM- oder Harddisks oder auch eine neue Hardcopyroutine installieren. Da das neue Programm meist nur einen Teil der Aufgaben der alten Routine übernimmt (z.B. nur die Operationen auf ein bestimmtes Laufwerk), muß es sich die Adresse des alten Unterprogramms merken und dieses gegebenenfalls aufrufen. Beim XBRA-Verfahren wird nun dem Einsprungpunkt in die neue Routine (also der Adresse, die in die Systemvariable eingetragen wird) folgende Struktur vorangestellt:

```
XB_MAGIC:     dc.l   "XBRA"
XB_ID:   dc.l    "NAME"
XB_VEC: dc.l      0
ENTRY:  ...
```

Das Langwort *XB_MAGIC* enthält den magischen Wert $58425241 (="XBRA"), um ein Programm nach dem XBRA-Standard überhaupt erkennen zu können. In *XB_ID* ist ein vier Byte langer, programmspezifischer Name enthalten. Bei der Installation merkt sich das Programm die Adresse der alten Routine in *XB_VEC* und schreibt in die betreffende Systemvariable die Adresse der neuen Routine (*ENTRY*). Da nun die Adresse der alten Routine direkt vor dem Einsprungpunkt in die neue steht, kann man, falls mehrere Programme nacheinander installiert wurden (z.B. RAM-Disk, ROM-Disk, Harddisktreiber...), leicht eine ganze Kette von Routinen zurückverfolgen. Es ist nun auch möglich, gezielt ein Programm aus dieser Kette zu entfernen und genau dies kann *XBRA.TOS*.

Kernstück des Programms ist die Funktion *getxbra()*. Man übergibt ihr in *vec* einen Zeiger auf den Zeiger auf die Routine (auf Deutsch: die Adresse der Systemvariablen). Der Inhalt der Systemvariablen und ihre Adresse werden zunächst auf Plausibilität geprüft, der Zugriff auf eine nichtexistierende oder ungerade Adresse würde nämlich zum Absturz führen. Sollte die in die Systemvariable eingetragene Routine dem XBRA-Standard entsprechen, so liefert die Funktion 1 zurück und legt in *name* die Programmkennung und in *next* die Adresse des nächsten Routinenzeigers (*XB_VEC*) ab, sonst wird 0 zurückgeliefert. Beim nächsten Aufruf von *getxbra()* kann der Wert von *next* wieder in vec übergeben werden, um die XBRA-Kette weiter zu verfolgen. Ein Beispiel für dieses Vorgehen finden Sie in der Funktion *prnpage()*. *un_link()* entfernt aus allen XBRA-Ketten die Routinen mit der Programmkennung id. Sollte dieselbe Kennung mehrmals in der Kette auftauchen, wird nur der erste Eintrag (d.h. das zuletzt installierte Programm) entfernt. Nicht alle Systemvektoren stehen an einer festen Adresse, manchmal muß diese auch erst erfragt werden. Ein Beispiel dafür findet sich in *varinit()*.

Nach dem Start zeigt das Programm auf mehreren Bildschirmseiten die XBRA-Ketten für eine Reihe von wichtigen Systemvektoren an. Das Umschalten der Seiten ist durch direkte Eingabe der Seitennummer möglich. Will man nun ein Programm "aushängen", so kann man nach der Anwahl der Option "R" die Programmidentifikation eingeben (Klein- und Großbuchstaben werden unterschieden). Nach einer Rückfrage entfernt das Programm die Routinen mit dieser ID aus allen Ketten. Mit der Escape-Taste können Sie das Programm beenden.

XBRA.TOS wurde mit TURBO C entwickelt. Die einzige dabei verwendete Besonderheit dieses Compilers ist die Definition des Typs *ADDR* als Zeiger auf einen Voidzeiger. Zeiger auf Objekte vom Typ *void* bezeichnen im ANSI-Standard Zeiger auf beliebige Objekte. Sie können eigentlich auch jeden anderen Zeigertyp verwenden. Auf die Verwendung von Prototypen habe ich bewußt verzichtet, da diese von den meisten Compilern für den ST sowieso nicht unterstützt werden. TURBO C-Benutzer können die Warnungen des Compilers einfach ignorieren. Da das Zeigergefummel in C nicht jedermanns Sache ist, habe ich, um die Übertragung in andere Sprachen zu erleichtern, die Funktion *getxbra()* auch in OMIKRON.BASIC implementiert. Beachten Sie bitte, daß Zugriffe auf die Systemvariablen unbedingt im Supervisormodus erfolgen müssen. (Bei Peeks und Pokes in OMIKRON.BASIC ist das immer der Fall.) In der BASIC-Version habe ich mir

übrigens die Plausibilitätsprüfung gespart, der Interpreter fängt Systemfehler ohne Absturz ab.

Es bieten sich einige Erweiterungsmöglichkeiten an. Zum einen sollte die Liste der Systemvektoren noch wesentlich ergänzt werden (z.B. um die VBL-Slots). In [1] und [2] werden Sie dazu viele Anregungen finden. Zum anderen könnte man in der Funktion *prnpage()* am Ende der ID-Kette die Adresse der letzten Routine (**vec*) ausgeben. Liegt dieser Wert im Bereich von $FC0000 bis $FEFFFE, so endet die Kette im ROM. Ist dies nicht der Fall, so ist das meist ein Hinweis auf ein Programm im Speicher, das nicht dem XBRA-Standard entspricht (immmer vorausgesetzt, Sie verwenden wirklich ein ROM-TOS). Wenn Sie nur noch XBRA-Programme verwenden, so haben Sie damit sogar die Möglichkeit, Viren in Ihrem System zu erkennen.

Sollte Ihr ATARI bis zum letzten Byte mit residenten Utilities vollgestopft sein und XBRA zeigt nach dem Start dennoch keine einzige Kennung an, so heißt das nicht unbedingt, daß das Programm nicht funktioniert. Tatsächlich verwenden bisher nur sehr wenige Programme das XBRA-Verfahren. Außerdem versteckt ein einziges nicht dem Standard entsprechendes Programm alle Routinen, die vorher auf demselben Systemvektor installiert wurden. Noch ein Hinweis: Wenn Sie ein Programm aushängen, so wird es durchaus nicht aus dem Speicher entfernt. Klinken Sie also etwa einen RAM-Disk-Treiber aus, so können Sie zwar nicht mehr auf die Daten zugreifen, den belegten Speicher bekommen Sie aber trotzdem nicht zurück. Somit ist der praktische Wert dieses Programms eher gering, es soll vielmehr die Möglichkeiten des XBRA-Verfahrens demonstrieren.

Andreas Kohler

Literatur:

[1] *Jankowski/Reschke/Rabich: ATARI ST Profibuch, Sybex, 1987*

[2] *A. Esser: Die Systemvariablen des TOS, ST-Computer 11 & 12/88*


```
  1:  /* XBRA.TOS
  2:   * Programm zur Anzeige von XBRA-Programmen
  3:   * im Speicher und zum gezielten Deaktivieren
  4:   * einzelner Programme.
  5:   * Entwickelt mit Turbo C 1.0, Andreas Kohler '89
  6:   * (c) MAXON Computer 1989
  7:   */
  8:
  9:  #include <stdio.h>
 10:  #include <tos.h>
 11:  #include <string.h>
 12:  #include <ctype.h>
 13:
 14:  /* Bildschirmsteuerung */
 15:  #define Cls()          printf("\033E")
 16:  #define Nowrap()       printf("\033w")
 17:  #define Top()          printf("\033H\033l")
 18:
 19:  typedef void **ADDR;
 20:
 21:  typedef struct {
 22:      char     xb_magic[4];
 23:      char     xb_id[4];
 24:      void     *xb_vec;
 25:  } XBRA;
 26:
 27:  typedef struct {
 28:      char     name[13];
 29:      ADDR     addr;
 30:  } SYSVAR;
 31:
 32:  int varnum;
 33:  SYSVAR vars[] = {
 34:      {"midivec", 0l},          {"vkbdvec", 0l},
 35:      {"vmiderr", 0l},          {"statvec", 0l},
 36:      {"mousevec", 0l},         {"clockvec", 0l},
 37:      {"joyvec", 0l},           {"midisys", 0l},
 38:      {"ikbdsys", 0l},          {"Trace", 0x24l},
 39:      {"Line A", 0x28l},        {"Line F", 0x2Cl},
 40:      {"Gemdos", 0x84l},        {"Bios", 0xB4l},
 41:      {"Xbios", 0xB8l},         {"etv_timer",0x400l},
 42:      {"etv_critic", 0x404l},   {"etv_term", 0x408l},
 43:      {"resvector", 0x42Al},    {"hdv_init", 0x46Al},
 44:      {"swv_vec", 0x46El},      {"hdv_bpb", 0x472l},
 45:      {"hdv_rw", 0x476l},       {"hdv_boot", 0x47Al},
 46:      {"hdv_mediach", 0x47El},{"exec_os", 0x4FEl},
 47:      {"dump_vec", 0x502l},     {"prt_stat", 0x506l},
 48:      {"prt_vec", 0x50Al},      {"aux_stat", 0x50El},
 49:      {"aux_vec", 0x512l},      {"", 0l}
 50:  };
 51:
 52:
 53:  void varinit()
 54:  {
 55:      ADDR     lp;
 56:      SYSVAR *sv;
 57:
 58:      sv = vars;
 59:      lp = (ADDR) Kbdvbase();
 60:      for (; ! sv->addr; sv++)
 61:          sv->addr = lp++;
 62:  }
 63:
 64:  int getxbra(vec, name, next)
 65:  ADDR vec;
 66:  char *name;
 67:  ADDR *next;
 68:  {
 69:      XBRA *xbp;
 70:
 71:      if ((long) vec % 2l)     /* ungerade Adresse */
 72:          return (0);
 73:      if ((long) *vec % 2l)
 74:          return (0);
 75:      if ((long) *vec < sizeof(XBRA))
 76:          return (0);
 77:      xbp = (XBRA *) *vec;
 78:      xbp--;
 79:      if (! strncmp(xbp->xb_magic, "XBRA", 4))
 80:      {
 81:          strncpy(name, xbp->xb_id, 4);
 82:          name[4] = '\0';
 83:          *next = &(xbp->xb_vec);
 84:          return (1);
 85:      }
 86:      return (0);
 87:  }
 88:
 89:  void un_link(id)
 90:  char *id;
 91:  {
 92:      register int i;
 93:      char name[5];
 94:      ADDR next, old;
 95:
 96:      for (i = 0; i < varnum; i++)
 97:      {
 98:          old = vars[i].addr;
 99:          while (getxbra(old, name, &next))
100:          {
101:              if (! strncmp(name, id, 4))
102:              {
103:                  *old = *next;
104:                  break;
105:              }
106:              old = next;
107:          }
108:      }
109:  }
110:
```

→

```
111:  void prnpage(n)
112:  int n;
113:  {
114:      register int i;
115:      ADDR next;
116:      char name[5];
117:
118:      Cls();
119:      printf(" 1... = Seitenwahl, R = Eintrag ");
120:      printf("entfernen, <ESC> = Abbruch ");
121:      for (i = 24 * n; i < varnum && i < (n+1) *
              24; i++)
122:      {
123:          printf("\n%12s  $%041X ",
124:                  vars[i].name, vars[i].addr);
125:          next = vars[i].addr;
126:          while (getxbra(next, name, &next))
127:              printf("%4s ", name);
128:      }
129:  }
130:
131:  main()
132:  {
133:      long oldstack;
134:      int page, c;
135:      char rmid[5];
136:
137:      Nowrap();
138:      oldstack = Super(NULL);
139:      varinit();
140:      for (varnum = 0; vars[varnum].addr; varnum++)
141:          ;
142:      prnpage(page = 0);
143:
144:      while ((c = toupper((int) Cnecin())) != 27)
145:      {
146:          if (c >= '1' && c <= '9')
147:          {
148:              if ((c - '1') * 24 <= varnum)
149:              {
150:                  page = c - '1';
151:                  prnpage(page);
152:              }
153:          }
154:          else if (c == 'R')
155:          {
156:              Top();
157:              printf(" Entfernen: ");
158:              scanf("%4s", rmid);
159:              Top();
160:              printf(" \'%s\' entfernen (J/N) ? ",
                          rmid);
161:              if (toupper((int) Cnecin()) == 'J')
162:                  un_link(rmid);
163:              prnpage(page);
164:          }
165:      }
166:      Super((void *) oldstack);
167:  }
```

```
 1:  ' getxbra() in OMIKRON.BASIC
 2:  ' Vec%l, Name$ und Nxt%l sind global, da
 3:  ' Rückgabeparameter in Fkt. nicht erlaubt
 4:
 5:  DEF FN Getxbra%
 6:    LOCAL L%L,I%
 7:    L%L= LPEEK(Vec%L)
 8:    IF ( LPEEK(L%L-12)=$58425241) THEN
 9:      Name$=""
10:      FOR I%=-8 TO -5
11:        Name$=Name$+ CHR$( PEEK(L%L+I%))
12:      NEXT I%
13:      Nxt%L=L%L-4
14:      RETURN 1
15:    ENDIF
16:  RETURN 0
```

PC-SPEED
MS-DOS* Hardware Emulator
für ATARI ST

COMDEX-Fall / Las Vegas / ATARI-Stand — Nov. '89



Von links nach rechts:

Hans-Jörg Sack
Entwickler von PC-SPEED

Theo Breuers
Compo Software GmbH
Int. Vertrieb für PC-SPEED

Jack Tramiel
ATARI Boss

Uwe Heim
Heim-Verlag
Generalvertrieb PC-SPEED

Warum nicht nur 'ATARI-Boss Jack Tramiel' von PC-SPEED so begeistert ist!

1. Die saubere Lösung

PC-SPEED belegt keinen Port (besserer Anschluß für andere Geräte). Der Schreibtisch bleibt frei. Keine umständlichen Zusatzgeräte nötig.

2. Komplette Speichernutzung (EMS)

Ihr ATARI ST hat mehr Speicher, nutzen Sie Ihn!
Durch PC-SPEED können Sie jetzt den kompletten Speicherbereich Ihres ATARI ST nutzen. Auf dem 1040 ST z.B. **704 KB**, beim MEGA ST2 und MEGA ST4 besteht die Möglichkeit, über den kompletten verbleibenden Speicher über schnelle RAM-Disk's zu verfügen.
Ein MUSS für alle, die mit weniger nicht zufrieden sind!

3. Ihr ATARI hat Ton

Ohne Ton haben Sie sehr viel weniger!

4. Grafik

PC-SPEED unterstützt neben CGA natürlich auch **HERCULES, SUPER HERCULES** (Hyperscreen — s. ST-Magazin / 68.00045 — 5/89), **OLIVETTI** und **ATT.**
5 Grafikemulatoren mit PC-SPEED.
Warum mit weniger zufrieden geben?

5 Die ATARI Maus-Unterstützung

PC-SPEED jetzt mit neuen MAUSTREIBERN.
— Benutzen Sie Ihre ATARI Maus beim PC-Programm oder
— die serielle Maus bei PC- und ATARI Programmen
Entscheiden Sie selbst,
ob Sie mit weniger zufrieden sind!?

6. Einfacher Einbau

Durch den einfachen Einbau (Sockel löten, Platine aufstecken) ist es jederzeit möglich, beim Kauf eines neuen ATARI ST-Modells, Ihren PC-SPEED mitzunehmen.

7. PC-SPEED und Festplatten

PC-SPEED läuft mit Festplatten von ATARI und VORTEX und Eickmann und CSI und Profile und . . . und . . . und . . .
Tausende von PC-SPEED Besitzern sind zufrieden, schließen Sie sich an!

8. Händler und Hotline

Über 350 Händler in Deutschland betreuen Sie auch nach dem Kauf. Und wenn Sie spezielle Fragen zu PC-SPEED haben, steht Ihnen als registriertem Kunden die HOTLINE beim Heim-Verlag zur Verfügung. PC-SPEED wird inzwischen in über 20 weiteren Ländern von Händlern vertrieben.

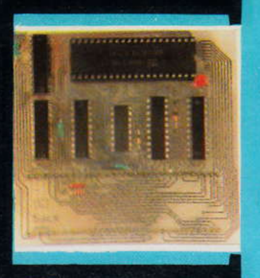
9. Das Buch zu PC-SPEED

Fast 300 Seiten von anerkannten Autoren zu den Themen
PC-SPEED Know how
— Perfekte Installation
— Ein MS-DOS* Lehrgang
— Tips und Tricks
Sämtliche PC-SPEED Händler in Deutschland zum Stand des 1.12.89 enthalten.

10. Immer Aktuell durch UP-TO-DATE

PC-SPEED ist eine **ganz offene** Lösung. Allein durch Software ist **vielfältige Erweiterung** möglich. PC-SPEED wird durch den Entwickler H.-J. Sack ständig erweitert. Durch Updates werden PC-SPEED Besitzer immer auf den neuesten Stand gesetzt. Was Sie heute noch wünschen, kann morgen schon gelöst sein. Haben Sie Wünsche zur Erweiterung des PC-SPEEDs, dann schreiben Sie uns. Der Entwickler überprüft die Realisierungsmöglichkeit.
PC-SPEED — wenn Sie mit weniger nicht zufrieden sind!

Wollen Sie sich mit weniger zufrieden geben?

BEGEIST  RUNG

Über 10.000 ATARI ST-Besitzer haben sich für PC-SPEED entschieden und täglich werden es mehr ...

Ihr Händler

Deutschland:
Über 350 Fachhändler beraten und betreuen Sie, bauen PC-SPEED ein.
Schweiz:
DataTrade AG, Zürich
Österreich:
Darius, Wien
Frankreich:
Upgrade Editions, Paris
England (UK):
Gasteiner Technologies, London
Niederlande:
ATARI (Benelux) B.V., Vianen
USA:
Michtron, Pontiac
Italien:
Eurosoft, Florenz
Schweden:
Spanien:
Kanada:
Micro D Distributors, Weston, Ontario
Jugoslawien:
Presdrag DRK, Zagreb

Internationaler Vertrieb:
COMPO SOFTWARE
P.O. Box 20
6269 ZG Margraten (NL)
Tel./Fax: 04458-2762
Niederlande

NEU: Das Buch

Schneider/
Steinmeier

PC-SPEED
Know how

* Perfekte Installation
* MS-DOS auf dem ATARI ST
* Tips u. Tricks

Alles, was Sie zum PC-SPEED wissen wollen.

Ca. 300 Seiten

Das richtige Geschenk.
Jetzt bestellen!

NEU: Das UPDATE





GANZ NEU —
Version 1.3
ab 15. Dezember 1989

1. Unterstützung der **ATARI ST-Maus**
2. **EMS — Extended Memory Specification**
   Durch **EMS** wird die übliche DOS-Speichergrenze von 640/704 KB gesprengt. Jetzt können Sie endlich den kompletten Arbeitsspeicher des ATARI ST-Computers nutzen. Z.B. können jetzt RAM-Disks bis zu 3 MB Kapazität (beim Mega ST4) im EMS-Speicher angelegt werden.
3. Ein residenter **HOTKEY** zur schnellen Farbmanipulation. Beim Drücken einer best. Tastenkombination kann der Bildschirm jetzt schnell **invers** geschaltet werden.
4. Neben der standardmäßigen CGA, HERCULES und OLIVETTI Grafikemulation kann auch der weit verbreitete **ATT-Monochrommodus** mit 640 x 400 Pixeln dargestellt werden.
5. WINDOWS läuft nun im OLIVETTI-Modus in hoher Auflösung auf dem ST.
6. Und weitere Verbesserungen.

Bestellen Sie die neue Update Version noch heute. Die Auslieferung erfolgt sofort nach Bestelleingang.
Benutzen Sie untenstehenden Bestell-Coupon.
Oder wenden Sie sich an Ihren zuständigen Fachhändler.

MS-DOS* für ATARI ST-Besitzer war gestern noch ein Traum von morgen. Heute nicht mehr, dank **PC-SPEED!** (*MS-DOS ist ein Warenzeichen der Microsoft Corp.)

Vertrieb weltweit:

Heim Verlag

Heidelberger Landstraße 194
6100 Darmstadt-Eberstadt
Telefon 0 61 51 - 5 60 57

BESTELLCOUPON

_____ **Update Version à 10,— DM**
(Für Diskette, Verpackung, Versand). Bitte 10,— DM als Geldschein oder Scheck im Briefumschlag beilegen.

_____ PC-SPEED **Know how**
Buch mit ca. 300 Seiten à 34,— DM
Zahlung [ ] per Scheck [ ] per Nachnahme, Versandkostenkosten zuzügl. 5,— DM unabhängig von bestellter Stückzahl

_____ Info Broschüre zu PC-SPEED mit Händlernachweis kostenfrei. Verwenden Sie auch die Bestellkarte in ST-Computer

Name: ______________________________

Straße: ______________________________

PLZ/Ort: ______________________________

Programmierte Logik

Teil 3: Der GAL-Prommer

Im abschließenden Teil unserer Serie über GALs stellen wir Ihnen ein Selbstbauprojekt für einen GAL-Prommer vor. Damit können Sie Ihre in den letzten Folgen erworbenen Kenntnisse in der Praxis anwenden und mit diesen Bausteinen experimentieren. Außerdem wird gezeigt, wie man PALs mittels GALs emulieren kann, d.h. in der Schaltung werden GALs statt PALs benutzt.

GALs als PALs

Die beiden GAL-Typen 16v8 und 20v8 können jeweils 21 PAL-Typen nachbilden (Tabelle 1). Hierzu ist nur ein geringer Aufwand nötig. Da bei den GALs genauso wie bei den PALs mit JEDEC-Dateien gearbeitet wird, kann man die bestehenden Dateien für PALs direkt übernehmen. In der dortigen Fuse-Liste befinden sich allerdings nur Angaben für die Logik-Matrix. Das Architektur-Control-Word, welches die Funktion der Ausgangszellen bestimmt, muß von Hand eingegeben werden. Hierzu laden Sie die vorhandene PAL-Datei in einen ASCII-Editor und schreiben das benötigte ACW dazu. In Bild 1 (für GAL16v8) und 2 (GAL20v8) sind einige ACWs aufgeführt.

Worin besteht nun der Unterschied zwischen einem PAL und einem GAL? Die PAL-Bausteine haben eine feste Beschaltung. Dadurch muß man sich vor dem Kauf sicher sein, wieviel Ein- und Ausgänge man benötigt, und welche Ausgangskonfiguration (normale oder inverse Ausgänge, Ausgänge mit Register...) der Baustein haben muß. Außerdem können PALs, wenn sie einmal gebrannt sind, nicht wieder gelöscht werden. Ergeben sich also später Änderungen [innerhalb der Logik oder auch an der Bausteinspezifikation (inverse statt normale Ausgänge)], muß ein neues PAL gekauft werden.

Diese und die in den bisherigen Folgen aufgezählten Nachteile haben die Firma Lattice wohl dazu bewogen, die GALs zu erfinden. Damit sich die Bausteine schneller durchsetzen können, und der Anwender seine bisherige Software (PAL-Assembler) weiter verwenden kann, hat man die GALs aufwärtskompatibel zu den vorhandenen PAL-Bausteinen gemacht.

Der Name der PALs setzt sich aus drei Teilen zusammen. An erster Stelle steht die Anzahl der Eingänge und Rückführungen auf die Fuse-Matrix. Danach folgt ein Kennbuchstabe. *H* steht für normale, *L* für inverse, *P* für normale Ausgänge mit Freigabe durch den ersten Produktterm, *R* für inverse Ausgänge mit Register und *RP* für normale Registerausgänge. Die letzte Angabe ist die Anzahl der Ausgänge, die der Baustein hat (bei den *R*- und *RP*-Typen ist es die Anzahl der Ausgänge, die Register haben). Aus diesen Angaben ergibt sich bereits ein Teil des ACW bei der Emulation. Bei den Bausteinen mit normalen Ausgängen müssen die XOR-Bits alle 1, bei inversen Ausgängen 0 sein. Die übrigen Bits für die Produkttermfreigabe und die Konfiguration der Ausgangszelle sind bausteinspezifisch und können Bild 1 und 2 entnommen werden.

GAL16v8:	GAL20v8:
10L8, 10H8, 10P8	14L8, 14H8, 14P8
12L6, 12H6, 12P6	16L6, 16H6, 16P6
14L4, 14H4, 14P4	18L4, 18H4, 18P4
16L2, 16H2, 16P2	20L2, 20H2, 20P2
16R8, 16RP8	20R8, 20RP8
16R6, 16RP6	20R6, 20RP6
16R4, 16RP4	20R4, 20RP4
16L8, 16H8, 16P8	20L8, 20H8, 20P8

Tabelle 1: PAL-Typen, die mit den GALs nachgebildet werden können

Die GAL-Prommer-Hardware

Der Schaltplan des GAL-Prommers ist in Bild 3 zu sehen, die dazugehörige Stückliste in Bild 4, das Platinen-Layout und der Bestückungsaufdruck in den Bildern 5 und 6.

Beim Aufbau der Hardware ist darauf zu achten, daß der Textool-Sockel und die beiden Leuchtdioden von der Lötseite bestückt werden. Am besten fangen Sie mit den niedrigsten Bausteinen (Widerständen) an. Bestücken Sie dann der Größe nach die weiteren Bauteile. Zum Schluß legen Sie den Textool-Sockel und die beiden Leuchtdioden in die dafür vorgesehenen Bohrungen des Gehäuses, stecken die Platine in den Gehäusedeckel und verlöten dann den Sockel und die LEDs. Wenn Sie die Platine fertig be-

```
10L8 und 10H8:
PTF:  11000000 11000000 11000000 11000000
      11000000 11000000 11000000 11000000
AC1:  00000000, SYN: 1,   AC0: 0

12L6 und 12H6:
PTF:  00000000 11110000 11000000 11000000
      11000000 11000000 11110000 00000000
AC1:  10000001, SYN: 1,   AC0: 0

14L4 und 14H4:
PTF:  00000000 00000000 11110000 11110000
      11110000 11110000 00000000 00000000
AC1:  11000011, SYN: 1,   AC0: 0

16L2 und 16H2:
PTF:  00000000 00000000 00000000 11111111
      11111111 00000000 00000000 00000000
AC1:  11100111, SYN: 1,   AC0: 0

16R8 und 16RP8:
PTF:  11111111 11111111 11111111 11111111
      11111111 11111111 11111111 11111111
AC1:  00000000, SYN: 0,   AC0: 1

16R6 und 16RP6:
PTF:  11111111 11111111 11111111 11111111
      11111111 11111111 11111111 11111111
AC1:  10000001, SYN: 0,   AC0: 1

16R4 und 16RP4:
PTF:  11111111 11111111 11111111 11111111
      11111111 11111111 11111111 11111111
AC1:  11000011, SYN: 0,   AC0: 1

16L8 und 16H8:
PTF:  11111111 11111111 11111111 11111111
      11111111 11111111 11111111 11111111
AC1:  11111111, SYN: 0,   AC0: 1
```

Bild 1: ACWs für das GAL16v8

```
14L8 und 14H8:
PTF: 11110000 11000000 11000000 11000000
     11000000 11000000 11000000 11110000
AC1: 00000000, SYN: 1,   AC0: 0

16L6 und 16H6:
PTF: 00000000 11110000 11110000 11000000
     11000000 11110000 11110000 00000000
AC1: 10000001, SYN: 1,   AC0: 0

18L4 und 18H4:
PTF: 00000000 00000000 11111100 11110000
     11110000 11111100 00000000 00000000
AC1: 11000011, SYN: 1,   AC0: 0

20L2 und 20H2:
PTF: 00000000 00000000 00000000 11111111
     11111111 00000000 00000000 00000000
AC1: 11100111, SYN: 1,   AC0: 0

20R8 und 20RP8:
PTF: 11111111 11111111 11111111 11111111
     11111111 11111111 11111111 11111111
AC1: 00000000, SYN: 0,   AC0: 1

20R6 und 20RP6:
PTF: 11111111 11111111 11111111 11111111
     11111111 11111111 11111111 11111111
AC1: 10000001, SYN: 0,   AC0: 1

20R4 und 20RP4:
PTF: 11111111 11111111 11111111 11111111
     11111111 11111111 11111111 11111111
AC1: 11000011, SYN: 0,   AC0: 1

20L8 und 20H8:
PTF: 11111111 11111111 11111111 11111111
     11111111 11111111 11111111 11111111
AC1: 11111111, SYN: 0,   AC0: 1
```

Bild 2: ACWs für das GAL20v8

stückt haben, sollten Sie noch einmal eine Sichtkontrolle durchführen. Stimmen alle Bauteile? Keine kalten (schlecht leitenden) Lötstellen? Sind die IC-Sockel richtig eingelötet (Kerbe des Sockels muß mit der Kerbe des Bestückungsaufdrucks übereinstimmen)? Sind alle ICs richtigherum eingesteckt?

Wenn Sie keine Fehler entdeckt haben, müssen Sie noch die EDIT-Spannung auf genau 16,5V einstellen. Verbinden Sie dazu den GAL-Prommer mit Ihrem Rechner [Drucker- und Joystickport (Pin 7)]. ACHTUNG: Dazu muß Ihr Rechner unbedingt ausgeschaltet sein! Schalten Sie den Rechner ein und laden Sie die Software. Wählen Sie den Punkt Brennen. Solange die rote Leuchtdiode brennt, können Sie die Spannung zwischen den Pins 2 (beim GAL20v8) oder 3 (beim GAL16v8) und 12 (Masse) abnehmen. Eingestellt wird Sie mit dem Trimmpoti R17.

Die Software

Die abgedruckt Software stellt die Funktionen zur Bearbeitung von GALs in einer einfachen Version zur Verfügung. Sowohl dem Fertiggerät als auch dem Platinenset liegt eine Diskette mit einer GEM-gesteuerten Software bei. Diese beinhaltet auch einen Editor für die Fuse-Matrix, welcher aus Platzgründen in der einfachen Version nicht enthalten ist.

Da so die JEDEC-Dateien mit einem ASCII-Editor erstellt werden müssen, wurde in der vorliegenden Software auf die Prüfung der Check-Summen verzichtet. Alle Bearbeitungmöglichkeitebn für GALs (Löschen, Brennen, Sichern, Lesen, Vergleichen) sind jedoch enthalten.

Das Programm wurde mit Turbo C Version 1.1 erstellt und ist auf der Monatsdiskette 1/2 '90 ab Ende Januar erhältlich.

Funktion der Hardware

Für alle, die sich näher mit der Funktion der Hardware befassen oder sich selbst eine Steuer-Software in einer anderen Programmiersprache schreiben wollen, werde ich im folgenden kurz auf die Hardware eingehen.

Die ICs 2 und 3 sind serielle Ein- und parallele Ausgabebausteine. An D7 werden die Daten angelegt, die mit D5 (bei IC2) und D6 (bei IC3) in den Baustein getaktet werden. Mittels der STROBE-Leitung werden die eingelesenen Daten-Bits am Ausgang sichtbar. IC5 erzeugt aus der 5V-Spannungsversorgung die benötigten 16,5V, um die GALs in den EDIT-Modus zu bringen.

Die Aufgabe der ICs im Einzelnen:

IC2: Die ersten 6 Bits stellen die Adresse der Reihe dar, die aus dem GAL gelesen oder in das GAL geschrieben werden soll. Das 7. Bit wird nur beim GAL20v8 benötigt und enthält dort einen Teil der Reihenadresse.

IC3:
Das erste Bit schaltet die Versorgungsspannung für das GAL20v8 an und aus. Bit zwei bewirkt das gleiche für das GAL16v8. Die nächsten beiden Bits schalten die EDIT-Spannungen (GAL16v8: Bit 4, GAl20v8: Bit 3). Das Bit Nummer fünf wird gesetzt, wenn ein GAL programmiert werden soll. Bit sechs ist das letzte an diesem Baustein benutzte Ausgangs-Bit und bestimmt, ob am Pin 11 des Textool-Sockels Masse (GAL16v8) oder D2 (GAL20v8) anliegt.

Hinweis

Es ist sicher nicht allzuleicht, logische Gleichungen zu vereinfachen. Da dies aber zum Teil unumgänglich ist, um die Gleichung in ein GAl brennen zu können,

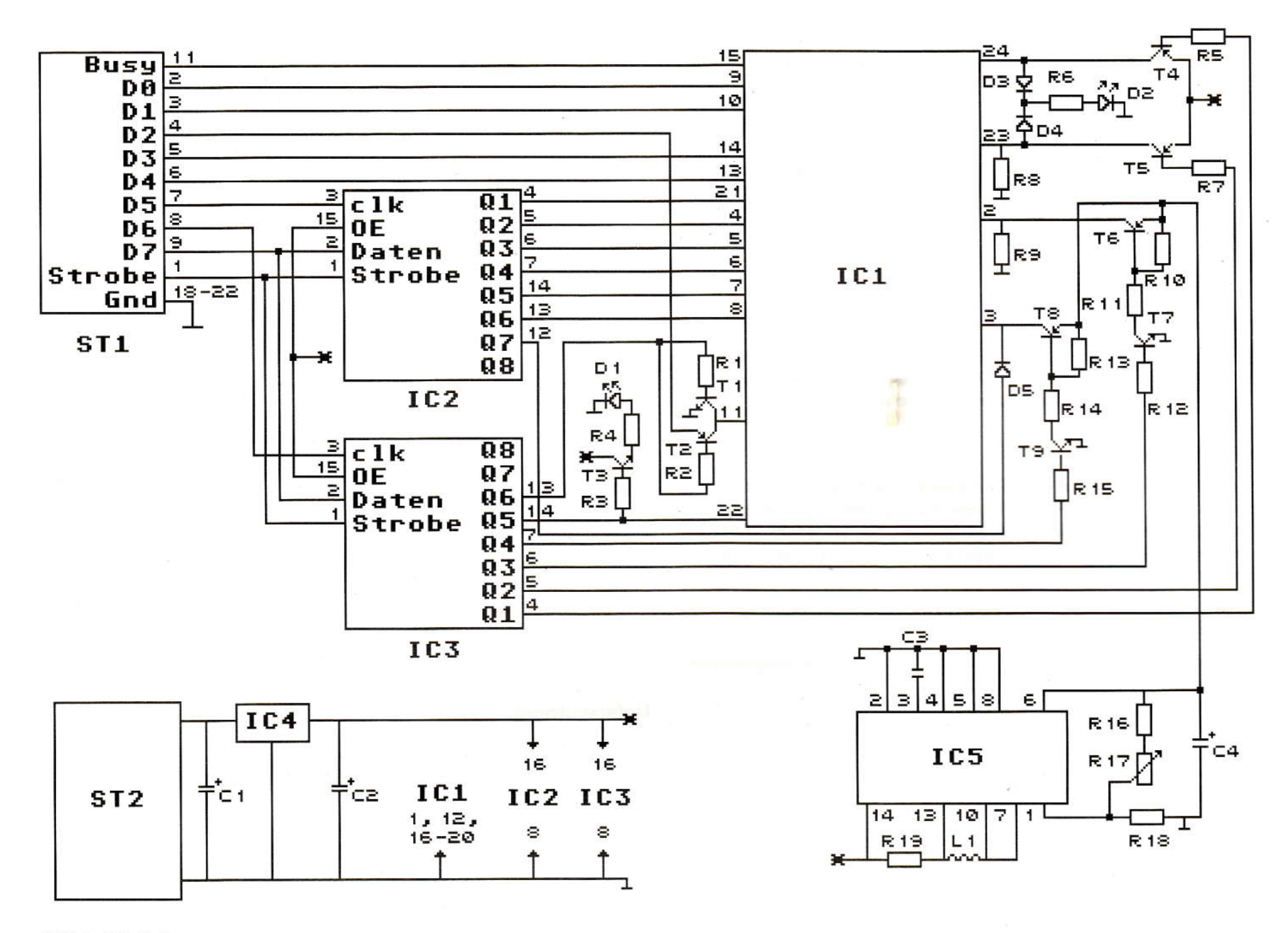


Bild 3: Schaltplan

möchte ich auf ein Programm hinweisen, das Ihnen diese Arbeit (zumindest zum Teil) abnimmt. Quinemac nennt es sich und ist als Sonderdisk bei MAXON zu beziehen. Außerdem werde ich in einer der folgenden Ausgaben in einem Grundlagenartikel auf die bekanntesten Verfahren zur Vereinfachung von logischen Gleichungen eingehen und kurze Beispielprogramme anführen.

Sollten Sie Anregungen oder Kritik zur GAL-Prommer Hard- oder Software haben, wenden Sie sich bitte an MAXON (Anschrift siehe Impressum).

Nachdem Sie nun die Grundlagen zur Programmierung sowie die benötigte Hard- und Software kennen, steht eigenen Experimenten mit diesen Bausteinen nichts mehr im Wege. Ich wünsche Ihnen dazu viele vergnügliche und lehrreiche Stunden.

Thomas Werner

Stückliste für GAL-Prommer V1.0:

Halbleiter:

IC1: 24 pol. Nullkraftsockel (schmal!)
IC2, IC3: 4094
IC4: 7805
IC5: TL 497

D1: LED grün "Programmieren"
D2: LED rot "Nicht öffnen"
D3, D4, D5: 1N4148

T1, T3, T7, T9: BC547 (o.ä.)
T2, T4, T5, T6, T8: BC557 (o.ä.)

L1: Drossel 100µH

Kondensatoren:

C1, C2: 100 µF, Elko, 16V
C3: 100 pF
C4: 100 µF, Elko, 25V

Widerstände
(1/4 W, 5%, Kohleschicht):

R1, R2, R8, R9,
R10, R11, R12,
R13, R14, R15: 10kΩ
R3, R5, R7, : 1kΩ
R4, R6: 180Ω
R16: 15kΩ, 1% Metall
R17: 1kΩ Poti
R18: 1,2kΩ, 1% Metall
R19: 1Ω

Sonstiges:

ST1: Doppelpfostenleiste, 26 pol.
ST2: Stromversorgungsbuchse

Achtung:
Bei Versorgung über die Joystickbuchse (Pin7) entfallen die Bauteile IC4, D6 und C1.
Der Nullkraftsockel (IC1) und die Dioden D1 und D2 werden von der Lötseite bestückt.

Bild 4: Stückliste

Literatur:

Lattice: GAL Handbook
mc 1/88: Programmierbare Logikbausteine
Monolithic Memories: PAL-Handbuch

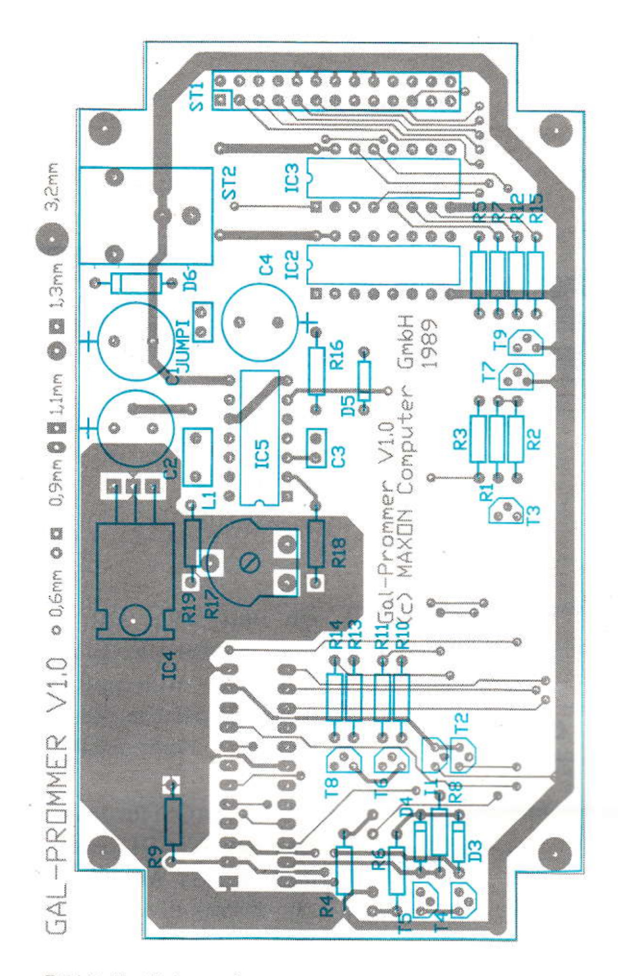


Bild 5: Bestückungsplan

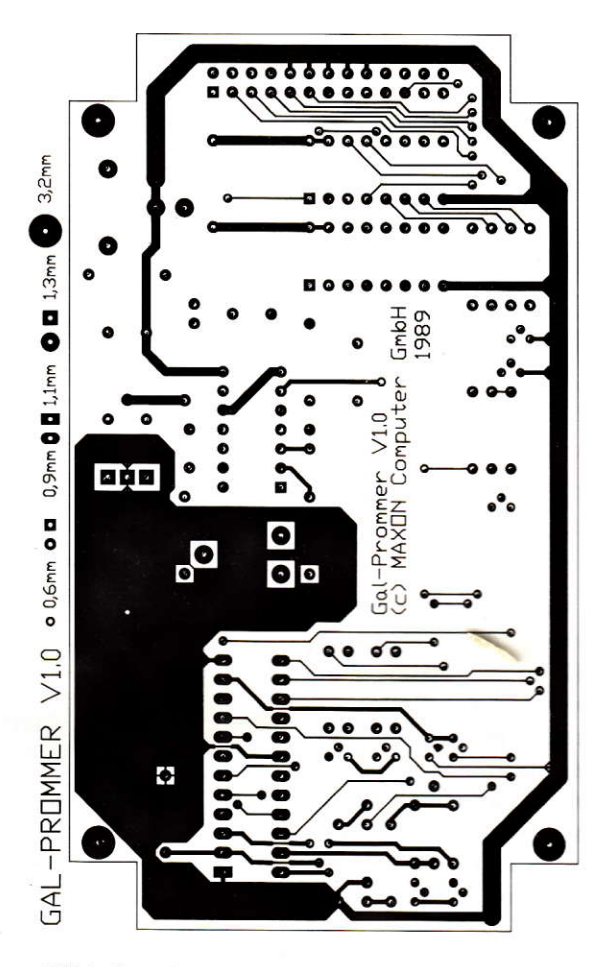


Bild 6: Bestückungsseite (1:1)

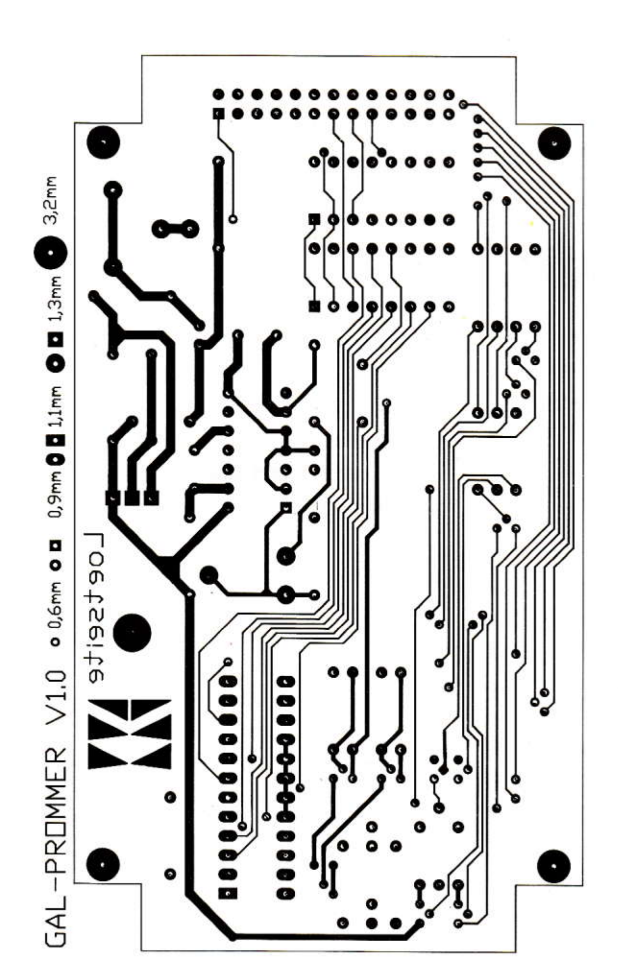
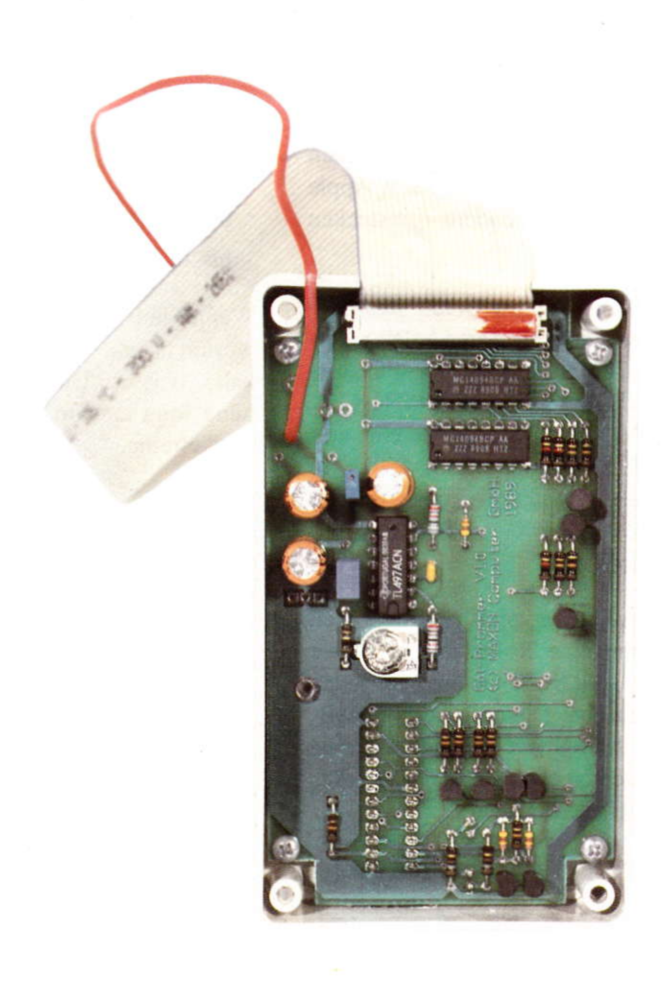


Bild 7: Lötseite (1:1)



Per SCSI zum ST

Das hat uns noch gefehlt. Während das Schlagwort 'SCSI' immer häufiger im Anzeigenteil der Fachpresse auftaucht, müssen ST-Besitzer, die nicht über das nötige Kleingeld für ein professionelles Fertiggerät verfügen, mit sogenannten Billiglösungen vorliebnehmen.

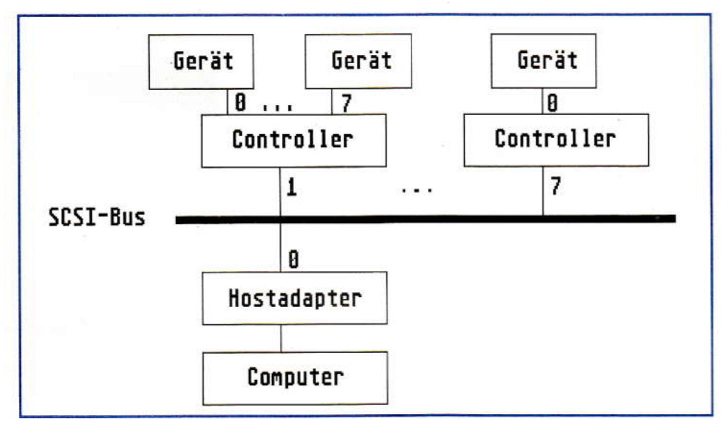


Bild 1: Beispiel eines SCSI-Systems

Damit ist es nun vorbei. Ein Host-Adapter für den Selbstbau eröffnet Ihnen die Welt der SCSI-Geräte. Festplatten beliebiger Größe, Wechselplatten, Streamer, CD-ROM etc. können bald Ihren Arbeitstisch bevölkern. Und sollten Sie einmal aufsteigen (z.B. ATARI TT), brauchen Sie Ihre Geräte nicht mehr für'n Apple und'n Egg zu verschleudern - umstecken genügt.

Wer da behauptet, diese Geräte seien viel zu teuer, soll eines Besseren belehrt werden: Eine 20MB-Festplatte in 'PC-Ausführung' (DM 440.-) plus OMTI-Controller (DM 170.-) ist teurer als eine entsprechende SCSI-Platte (DM 580.-). In diese ist der Controller bereits eingebaut. Die Preise sind diversen Fachanzeigen entnommen.

Neben den Kosten spielt natürlich auch die Leistung eine Rolle, und in dieser Beziehung sind SCSI-Geräte über jeden Zweifel erhaben. Die preiswerte 20MB-Platte des Autors erreicht mühelos eine Transferrate von über 600 MB/s. Sie dürfen vergleichen, wir nicht.

Noch ein paar Worte zum Host-Adapter:

- unterstützt alle Kommandogruppen (Gruppe 0 bis Gruppe 7)
- Das angeschlossene Gerät ist uneingeschränkt bootfähig.
- handelsübliche Treiber ohne Änderung lauffähig
- gepufferter DMA-Bus
- bis zu vier SCSI-Geräte anschließbar

Wenn Sie bereits Ihren Lötkolben angeworfen haben, folgt jetzt der Sprung ins kalte Wasser: Die Hardware-Beschreibung folgt erst im nächsten Teil dieser Artikelreihe.

Erst die Arbeit ...

Wir werden uns zunächst mit dem SCSI-Bus und den Unterschieden zum ACSI-Bus beschäftigen. Diese Grundlagen sind die Voraussetzung, um die Beschreibung der Hardware zu verstehen, und ermöglichen die Realisierung spezieller SCSI-Software. Wenn Sie einen Drucker an den Centronics-Port anschließen, so verbinden Sie einen Sender mit einem Empfänger. Die Zuordnung Sender/Empfänger kann zwar wechseln, doch sind Sie nicht in der Lage, einen zweiten Drucker anzuschließen und sich per Software einen davon auszusuchen. Dazu brauchen Sie ein Bussystem. Auf diesem werden außer den eigentlichen Daten stets auch Steuerinformation übertragen, die sozusagen den Verkehr regeln.

Das Wort ACSI für den DMA-Bus des ATARI dürfte den meisten von Ihnen geläufig sein. Dieses *A*TARI *C*omputer *S*ystem *I*nterface ist leider nur eine stark abgemagerte Version seines Ziehvaters SCSI (oder war es die Mutter?). Vielleicht gefiel unserem Mr. Tramiel das erste Wort 'Small' nicht? Der SCSI-Bus verbindet bis zu acht Controller miteinander. Diese Controller sind recht intelligente Burschen. Maximal acht Geräte kann (theoretisch) jeder davon verwalten. Nun macht es wenig Sinn, z.B. 64 Festplatten miteinander kommunizieren zu lassen, und darum befindet sich meist ein Computer, der sogenannte Host, unter den Geräten. Wenn der Computer keine Bevormundung durch einen Controller benötigt, wird er einfach mit Hilfe eines (jetzt kommt's) Host-Adapters in den Bus eingeklinkt.

Auf dem ACSI-Bus ist der ST eindeutig der Chef. Er ergreift die Initiative zur Datenübertragung und unterscheidet die Zielgeräte anhand ihrer Target-Nummer. Der SCSI-Bus ist da etwas flexibler. Jeder der angeschlossenen Partner kann Initiator spielen und ein beliebiges Ziel (engl.: Target) anwählen.

Die Phasen oder wie verwirre ich den Leser

Wie bereits erwähnt, passieren neben der Datenübertragung noch andere interes-

sante Dinge auf dem Bus. Will ein Gerät den Bus verwenden, zeigt es dies an, indem es die eigene Identifikationsnummer auf den Bus legt (SCSI-ID, entspricht etwa der Target-Nummer auf dem DMA-Bus).Wenn mehrere Geräte annähernd gleichzeitig den Bus verwenden wollen, gewinnt das mit der höchsten Nummer. In dieser *Arbitration-Phase* wird also der Initiator ermittelt.

Runde 2: Während der *Selection-Phase* legt der Initiator die SCSI-ID des Zielgerätes auf den Bus und wartet auf Antwort. Nachdem sich das Target gemeldet hat, kann es endlich losgehen. Alle anderen Geräte dürfen das Paar bis zum Ende der Übertragung nicht mehr stören.

Stellen Sie sich bitte vor, das Target sei ein Bandgerät und muß zur Datenübertragung zunächst vor- oder zurückspulen. Um den Bus nicht minutenlang zu blokkieren, kann das Bandgerät (besser gesagt sein Controller) den Bus freigeben, und später in der sogenannten *Reselection-Phase* den ursprünglichen Initiator anwählen und die Übertragung fortsetzen.

Solche Leckerbissen kann uns der ACSI-Bus leider nicht bieten, die folgenden Phasen dürften aber den meisten von Ihnen bekannt vorkommen. Die *Command-Phase* zur Übergabe des Kommandos und der Übertragungslänge sowie die *Data-Phase* (was macht die bloß?) ähneln ebenso ihrem ACSI-Pendant wie die *Status-Phase* zum Abschluß der Übertragung. Halt, da gibt es ja noch die *Message-Phase* beim SCSI-Bus. Während das Status-Byte das Ergebnis der Übertragung anzeigt (OK / Fehler), beendet erst das Messagebyte vom Target den Zugriff auf den Bus. Normalerweise wird eine '0' mit der Bedeutung 'Command Complete' übertragen, doch auch hier existiert eine Vielfalt, die den Rahmen dieses Artikels bei weitem sprengen würde. Dazu kommt noch, daß auch der Initiator Messagebytes senden kann ...

Der Ruhezustand nennt sich *Bus Free-Phase*. Sämtliche Signale sind inaktiv. Alle warten darauf, daß endlich was passiert.

ACSI goes SCSI

Solange wir noch nicht im Besitz eines ATARI TT sind, können wir auf solche Dinge wie Arbitration, Reselection und Messagebyte getrost verzichten - der ST kann's sowieso nicht. Dies widerspricht übrigens nicht der SCSI-Norm. Unser System nennt sich 'Single Initiator, Multi Target', d.h. der ATARI braucht sich nicht zu duellieren, um den Buszugriff zu erhalten. Er ist der einzige, der dazu überhaupt in der Lage ist. Alle anderen Geräte auf dem Bus sind Targets, die sich erst dann rühren, wenn sie dazu aufgefordert werden. Dadurch entfällt Arbitration, aber auch Reselection. Der gefürchteten Inkompatibilität zuliebe verzichten wir freiwillig auf das Messagebyte. Das soll gefälligst der Host-Adapter erledigen.

Wenden wir uns nun den Leitungen zu, auf denen all diese Vorgänge ablaufen. Invertierte (also low-aktive) Signale sind durch ein vorangestelltes '/' gekennzeichnet:

- /DB(0-7)
  Dies sind die Datenleitungen des SCSI-Busses. Im Gegensatz zum ACSI-Bus sind die acht Daten-Bits low-aktiv.

- /DB(P) Paritätsbit.
  Zum Glück bietet jeder SCSI-Controller die Möglichkeit, die Paritätskontrolle zu unterdrücken.

- /BSY Busy.
  Das Target kennzeichnet damit den Bus als belegt.

Schreiben Lesen
/REQ vom Target
/ACK vom Initiator
Daten
SCSI-Handshaking
/REQ vom Target
/ACK vom ATARI
250ns 250ns
Daten
ACSI-'Handshaking'

Bild 2: Handshaking im Vergleich

- /SEL Select
  wird vom Initiator während der Selektionsphase aktiviert.

- /CD Control/Data.
  Das Target unterscheidet damit zwischen Daten und Steuerinformationen (Kommando, Status).

- /IO Input/Output
  steuert die Datenrichtung und entspricht in seiner Funktion dem Read/Write-Signal auf dem ACSI-Bus. Sender ist das Target.

- /MSG
  Das Signal Message vom Target kennzeichnet die ________-Phase (bitte richtige Antwort eintragen).

- /REQ
  Das Target sendet Request, um die Übertragung eines Bytes zu beginnen. In der Data-Phase entspricht dies dem ACSI-Signal DRQ (Data Request).

- /ACKAcknowledge
  vom Initiator entspricht in der Data-Phase dem gleichnamigen ACSI-Signal. Es bestätigt die mit /REQ begonnene Übertragung.

- /RST Reset.
  Der Name spricht für sich.

Shake Hands

Die asynchrone Datenübertragung beruht auf dem Prinzip, daß der eine Partner nur aktiv wird, wenn der andere ebenfalls dazu bereit ist, und umgekehrt. (Das habe ich in einem anderen Zusammenhang auch schon mal gehört.) So können sich auch unterschiedlich schnelle Geräte aneinander anpassen. Da das Handshaking auf dem SCSI-Bus sehr konsequent durchgeführt wird, soll einmal die Übertragung eines Bytes vom Host (Initiator) zum Controller (Target) als Beispiel dienen.

Das Target aktiviert *REQ*, um seine Bereitschaft zum Empfang eines Bytes anzuzeigen. Mit den Daten setzt der Initiator auch *ACK* und kündigt damit die Daten an. Sobald das Target das Byte übernommen hat, wird *REQ* zurückgesetzt. Bedeutung: Die Daten sind angekommen. Nun kann der Initiator befriedigt sein *ACK* deaktivieren, worauf das Target mit einem neuen Zyklus beginnt.

Viele glauben, auch der ATARI verfolgt dieses Prinzip des Händeschüttelns, zu-

zumindest in der Datenphase. Das stimmt aber nur zur Hälfte. Um bei obigem Beispiel zu bleiben: Unser Rechner wartet tatsächlich auf das *REQ* des Targets, doch sein Acknowledge bleibt nur ca. 250 ns lang aktiv. Kurz danach verschwinden auch die Daten wieder - ob das Target sie inzwischen empfangen hat oder nicht! Dies ist auch eines der großen Probleme auf dem ACSI-Bus. Die Daten liegen für 250 + 2 * 60 = 370 ns an, nicht mehr und nicht weniger.

Für die umgekehrte Richtung existieren ähnliche Probleme. Will unser ST ein Byte lesen, erwartet er gültige Daten, solange sein *ACK*-Signal aktiv ist. Beim korrekten Handshaking, wie es auf dem SCSI-Bus realisiert ist, sind die Daten nur solange gültig, bis das Target sein *REQ* deaktiviert hat.

Details are Hacker's best friends

Doch zurück zum SCSI-Bus. Die Data Phase ist durch das oben beschriebene Handshaking bereits vollständig erklärt. Bis auf die dargelegten Differenzen verläuft die Übertragung auf dem ACSI-Bus identisch.

Will der Initiator ein Gerät selektieren, aktiviert er das Daten-Bit, welches der SCSI-ID des Empfängers entspricht. Soll also der Controller mit der Nummer 2 angesprochen werden, wird das Daten-Bit 2 auf low gesetzt und zwar ausschließlich. Nach einer Pause von mindestens 90ns wird Select aktiviert (für mathematisch und/oder technisch nicht so interessierte Leser: 90 Nanosekunden sind 0.00000009 Sekunden). Fühlt sich ein Target angesprochen, aktiviert es die

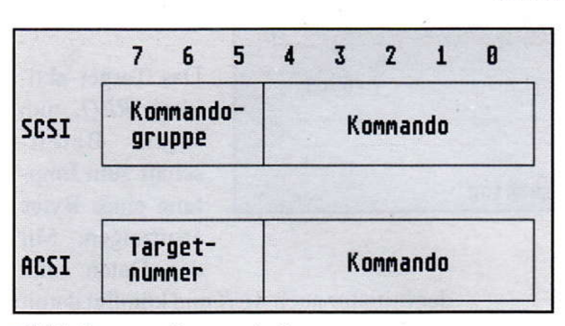


Bild 3: Das erste Kommando-Byte

Busy-Leitung. *Busy* bleibt bis zum Ende der Übertragung gesetzt. Nun ist die Verbindung hergestellt, und der Initiator nimmt das Datenbit und *SEL* zurück.

Von nun an übernimmt der Controller das Ruder. Alle (!) folgenden Bytes werden

Die Kommandos der Seagate-SCSI-Festplatten

Gruppe	Kommando	Opcode	Bezeichnung	Bemerkung
0	00	00	Test Unit Ready	
	01	01	Rezero Unit	
	03	03	Request Sense	
	04	04	Format Unit	
	07	07	Reassign Blocks	
	08	08	Read	
	0A	0A	Write	
	0B	0B	Seek	
	11	11	Read Usage Counter	
	12	12	Inquiry	
	15	15	Mode Select	
	16	16	Reserve	
	17	17	Release	
	1A	1A	Mode Sense	
	1B	1B	Start/Stop	
	1C	1C	Rec. Diag. Results	Nur ST225N
	1D	1D	Send Diagnostic	
1	05	25	Read Capacity	
	08	28	Read	
	0A	2A	Write	
	0B	2B	Seek	Nicht ST225N
	0F	2F	Verify	
	17	37	Read Defect Data	
	1B	3B	Write Buffer	Nicht ST225N
	1C	3C	Read Buffer	Nicht ST225N
7	05	E5	Read Long	Nicht ST225N
	06	E6	Write Long	Nicht ST225N

durch einen *REQ/ACK*-Handshake übertragen, so auch der 6- oder 10-Byte-Kommandoblock, der als nächstes folgt. Hier unterscheidet sich der ACSI-Bus wohl am meisten vom SCSI-Bus. Erstens hat ATARI die Selektionsphase in das erste Kommandobyte integriert. Zweitens bedient sich ATARI der separaten Handshake-Leitungen *CS* und *IRQ*. Dies ist allerdings notwendig, da drittens das Handshaking während der Kommandophase 'falschherum' arbeitet, also mit aktivem Initiator. Das Target antwortet nur. Dies ist zumindest die allgemeine Auffassung.

Man kann es auch anders sehen: Außer beim ersten Kommando-Byte fordert das Target die Daten mit *IRQ* an, und der Initiator antwortet mit *CS*. Aus dieser Sicht fällt die Umsetzung in ein SCSI-konformes Handshaking leichter.

Betrachten wir einmal das erste Kommando-Byte. Die Bits 0-4 bezeichnen das Kommando, welches das Gerät ausführen soll. Die Bits 5-7 selektieren auf dem ACSI-Bus eines der angeschlossenen Targets. SCSI-Geräte erkennen daran allerdings die gewählte Kommandogruppe. Selektiert sind sie zu diesem Zeitpunkt bereits. Einfache Host-Adapter übergeben dem Zielgerät einfach eine Null und beschränken den Anwender damit auf Gruppe 0-Kommandos. Wir werden später noch sehen, wie diese Beschränkung zu umgehen ist.

This is the end

Nach der Phase *Data-in* bzw. *Data-out* wird zunächst ein Status-Byte und anschließend ein Messagebyte zum Initiator übertragen. Auch hier wirkt wieder der inzwischen bekannte *REQ/ACK*-Handshake. Die Message-Phase ist am gesetzten *MSG*-Bit zu erkennen (auf low gesetzt, versteht sich). Diese kurze Beschreibung läßt schon erahnen, daß die Vorgänge auf dem SCSI-Bus durch ihre Gleichförmigkeit erheblich leichter zu verstehen sind als das ACSI-Verhalten. Der DMA-Bus hat selbst für das Status-Byte noch eine Variante parat: Es werden,

wie in der Command-Phase, die Signale *CS* und *IRQ* verwendet. Doch diesmal wird das Target aktiv und setzt zunächst IRQ auf low. Darauf wartet die Zeitschleife des Programmierers schon. Das Statuslesen komplettiert mit einem Impuls auf *CS* das Handshaking.

Im nächsten Beitrag geht es dann zur Sache. Wie kann ich diese unterschiedlichen Anforderungen vereinen? Warum muß der Entwickler unbedingt GAL-Bausteine verwenden? Wie komme ich an die neuen Kommandogruppen heran? Warum bohre ich mir kein Loch ins Knie und schütte Milch rein?

Reiner Wiechert

Literatur:

Brod/Stepper: Scheibenkleister II

Jankowski/Reschke/Rabich: ATARI ST Profibuch

SEAGATE Technology: ST125N/138N/157N Product Manual, Rev.C

SEAGATE Technology: SCSI Interface Manual, Rev.B

Scientific Micro Systems: SCSI Intelligent DataControllers, Reference Manual, Rev.C



TEIL 1: GRÜSSE AUS WALHALLA

Wodan, der Merkur, dem Gott des Handels in der römischen Mythologie, gleichgestellt ist, gab den Namen für ein kleines Accessory, welches eine Zeichenkette an die Tastatur übermittelt.

```
WODAN        Copyright 1989 by D. Rabich.
             Entwickelt mit Megamax Modula-2.        an  aus
1 Sehr geehrte Damen und Herren,
2 Es verbleibt mit freundlichen Grüßen
3 Mit freundlichen Grüßen
4 P.S.:
5 Heini Hubert Uderzostraße 4711 1234 Hintertupfingen
6 Schreib mal wieder!
7 Kennwort
8 Liebe Claudia! Vielen Dank für Dein Schreiben vom
9 Werter Wilhelm!
0 Tel. 054321 - 12345
   OK    Abbruch    Hilfe    laden    speichern
```

Bild 1: Wodan in Aktion

Ursprünglich sollte das Programm nach Hermes, in der griechischen Mythologie unter anderem der Götterbote, benannt werden, da das Accessory einem Boten der Tastatur eine Botschaft übergibt, aber ein Programm mit diesem Namen existiert schon. Dann hieß es Merkur, aber auch dieser Name war schon vergeben. Schließlich wurde aber doch ein Name gefunden: Wodan.

Die Aufgabe von Wodan ist es, um es auf "computer-deutsch" auszudrücken, im Tastatur-Puffer bestimmte zwischengespeicherte Tasten durch Zeichenketten zu ersetzen. Dabei soll das Programm diversen Anforderungen genügen: Es soll unter möglichst vielen Programmen wirksam sein, die Zeichenketten sollen jederzeit edierbar und von Programmen individuell austauschbar sein, verschiedene Sätze von Zeichenketten sollen abgespeichert und nachgeladen werden können. So die einfache Aufgabenstellung.

Programmtechnisch realisieren läßt sich dies, indem Wodan als Accessory betrieben wird, denn dann lassen sich die Zeichenketten jederzeit edieren. Die Wirksamkeit unter möglichst vielen Programmen wird erreicht, wenn der *IKBD*-Systemvektor "verbogen" wird und eine selten benutzte Tastenkombination zum Aufruf einer Wodan-internen Routine Verwendung findet. Sätze mit Zeichenketten lassen sich am besten dort nachladen bzw. abspeichern, wo auch die Zeichenketten ediert werden.

Und schon tauchen neue Probleme auf. Woher bekommt man den *IKBD*-Systemvektor (*IKBD* heißt übrigens **I**ntelligent-**K**ey**B**oar**D**)? Was muß beim Verbiegen des Vektors beachtet werden? Welche Tastenkombination wird selten benutzt? Wie kommen fremde Programme an die Zeichenketten?

Die Fragen lassen sich recht einfach beantworten. Die XBios-Funktion 34 - *Kbdvbase* - liefert einen Zeiger (oder Pointer) auf eine Struktur, die verschiedene Adressen auf diverse Routinen enthält. Addiert man zu der erhaltenen Adresse 32, findet man den *IKBD*-Systemvektor. Die anderen Adressen sollen uns hier nicht interessieren. Beim Verbiegen des Vektors bietet sich das sich allmählich durchsetzende *XBRA*-Verfahren an.

Das *XBRA*-Verfahren hat seinen Namen von "e**X**tended **BRA**ner" und die Idee dazu stammt von Moshe Braner. Das Verfahren ist recht einfach: Vor die Adresse, auf die der neue Vektor gesetzt wird, plaziert man die folgende (C-)Struktur:

```
typedef struct
{
  char xb_magic[4];  /* fest "XBRA" */
  char xb_id[4];     /* ID des installierten
                        Programms */
  long xb_oldvec;    /* ursprüngliche
                        Adresse */
} XBRA;
```

Der Vorteil dieses Verfahrens ist, daß man leicht erkennen kann, ob das Programm schon einmal installiert wurde. Außerdem läßt sich die Adresse der ursprünglichen Vektors jederzeit ermitteln.

Die Konstante *XBRA* muß immer in der Struktur aufgeführt sein, danach steht die Identity des installierten Programms, für Wodan ist dies *DRRH*. (Alle Kennungen mit den ersten drei Buchstaben *DRR* sind für den Autor von Wodan reserviert.) Wer eine ID für sein Programm haben möchte, der kann sich in die Liste eintragen lassen, die in der Mailbox Maus in Münster geführt wird.

Fremde Programme nutzen die *Message-Pipe*, um bei Wodan nach der Adresse der Zeichenketten zu fragen. Hierzu später mehr.

Es bleibt nur noch die Frage nach der Tastenkombination. *Alternate* ist eine Taste, die selten benutzt wird. Also bietet sich diese - in Verbindung mit einer anderen - an. Die Wahl der anderen Taste(n) fällt nicht schwer: A bis Z und die Ziffern 0 bis 9. Die 36 Möglichkeiten sollten reichen.

Die Aufgabenstellung und einige programmtechnische Anforderungen wären damit erledigt. Wenden wir uns nun der Umsetzung der Idee in ein Programm zu. Die Wahl der Programmiersprache beziehungsweise der Entwicklungsumgebung ist schnell erledigt. Es soll eine leserliche, selbstredende Programmiersprache sein, die jeder verstehen kann. Das Programm muß sich natürlich modularisieren lassen. Die Wahl fällt folglich auf Modula-2, in unserem Fall Megamax Modula-2.

Die Eingabe und Veränderung der Zeichenketten erfolgt in einer Dialogbox, wie sie auf der ersten Bild zu sehen ist. Zehn der 36 Zeichenketten können eingesehen werden, die restlichen erscheinen, wenn der Slider verschoben wird. Aber zu der Dialogbox später nochmal.

Der Programmaufbau - in Bild 2 dargestellt - zeigt die GEM-spezifischen Routinen in eigenen Modulen: *HandleWodan* übernimmt die eigentliche Verwaltung, Wodan enthält die Indizes der Objekte in den Dialogboxen, und GEMUtility beinhaltet diverse, recht allgemein gehaltene Routinen. Das Modul *XBRA*, welches Thomas Tempelmann zur Verfügung gestellt hat, sorgt für alles, was mit dem *XBRA*-Verfahren zu tun hat.

Das Modul *CommunicationWodan* sorgt für die Übermittlung der Adresse der Zeichenketten, stellt also die Schnittstelle zu fremden Programmen dar. Das wichtigste Modul - *WodanAcc* - hängt bei Bedarf eine neue Routine vor den *IKBD*-Systemvektor und übernimmt alle groben Programmsteuerungen.

Soweit die kurze Vorstellung des Programms Wodan. Die einzelnen Listings werden in diesem und den folgenden Heften vorgestellt. Heute beginnen wir mit *WodanAcc* und den Definitionsmodulen von *CommunicationWodan*, GEMUtility, *HandleWodan* und *XBRA*. Im nächsten Heft werden die Resource-Datei (und damit das Definitions- und das Implementationsmodul Wodan) erstellt und das Implementationsmodul *XBRA* vorgestellt. Im übernächsten Heft folgt das Implementationsmodul *HandleWodan* und in der letzten Folge die Implementationsmodule zu *CommunicationWodan* und *GEMUtility* sowie ein kleines C-Programm, welches zeigt, wie ein fremdes Programm bei Wodan die Zeichenketten verändern kann.

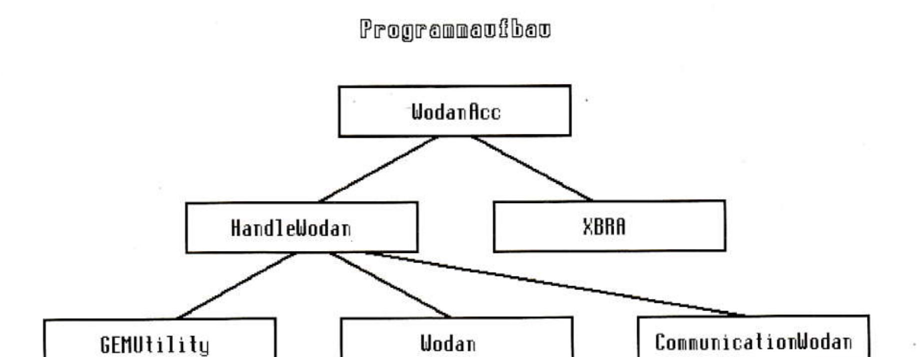


Bild 2: So agiert Wodan

Die Bedienung von Wodan ist einfach gehalten. Da Wodan als Accessory betrieben wird und sich in das Drop-Down-Menü *Desk* (mit dem Programmnamen, dem ATARI-Zeichen oder auch *Desk* als Titel) einträgt (siehe Bild 3), erreicht man die manipulierenden Funktionen wie Veränderung der Zeichenketten, wenn der Mauszeiger auf den Menütitel bewegt wird und dann weiter auf den Eintrag *Wodan....* Auf einen Klick mit der linken Maustaste hin öffnet sich eine Dialogbox mit den zentralen Funktionen von Wodan.

In dieser in Bild 1 dargestellten Dialogbox können die Zeichenketten ediert werden, wie es auch sonst in Dialogboxen üblich ist. Da immer nur 10 der 36 Zeichenketten dargestellt werden, ist eine Möglichkeit nötig, sich die anderen Zeichenketten anzeigen zu lassen. Dazu wird der Slider nach unten oder oben bewegt, womit ein anderer Ausschnitt der 36 Zeichenketten sichtbar wird. Ein Klick auf den Hintergrund unterhalb des Sliders bewirkt, daß ein Seite nach unten geblättert wird. Nach oben geht's natürlich analog. Klickt man einmal auf die Pfeile, wird eine Zeile nach unten oder oben gescrollt. Selbstverständlich kann man die Maustaste auch gedrückt halten, wodurch solange weitergescrollt wird, bis das Ende bzw. der Anfang der Liste erreicht ist. Zusätzlich zu diesen Möglichkeiten bewirkt ein Doppelklick auf einen der Pfeile, daß der Anfang bzw. das Ende der Liste sichtbar wird.

File Bearbeiten
1st Word Plus...
Kubis
Wodan...
Signum - Scrcop
LittleSweet
Disk-Utility

Bild 3: Wodan ist als Accessory dauernd erreichbar

Mit *an* bzw. *aus* wird Wodan an- bzw. ausgeschaltet. Da einige Programme die Alternate-Taste in Verbindung mit anderen benutzen, ist diese Abschaltmöglichkeit nötig. Sonst könnte es Probleme geben, aber die sollen ja vermieden werden. Ein Klick auf *Hilfe* wechselt zu einer Hilfe-Seite. *Laden* bzw. *Speichern* ermöglicht das Laden oder Abspeichern eines Parametersatzes. Diese Parameterdatei endet wie unter GEM-Programmen üblich auf die Extension *INF*. Die standardmäßig während des Bootvorgangs geladene Parameterdatei heißt folglich *WODAN.INF*, da das Programm den Namen *WODAN* trägt. Daraus resultiert, daß der volle Name des Programms oder besser des Accessories *WODAN.ACC* und der der Resource-Datei *WODAN.RSC* lautet. Da mehrere Parameterdateien benutzt werden können und diese vom Namen her in den vorgegebenen Rahmen passen müssen, liegt die Wahl von Namen wie *WODAN*.INF* nahe. Das Sternchen steht für zwei beliebig wählbare Buchstaben oder Ziffern. Soweit die Bedienung von Wodan.

Wodan verträgt sich in der vorliegenden Form mit fast allen Hardware-Konfigura-

tionen. Wer Wodan auf einem Monitor mit mehr als einer Farbebene verwenden möchte (also beispielsweise mit der Auflösung 640 mal 200 Pixel bei vier Farben), der muß die Resource-Datei ein wenig verändern. Betroffen ist lediglich das Icon *Wodan*, welches in der Größe angepaßt werden muß. Auf einem monochromen Monitor beliebiger Auflösung bereitet der Betrieb von Wodan keinerlei Schwierigkeiten. Einen Wermutstropfen gibt es allerdings doch. Die Verwendung von Wodan unter *GEM 2.x* ist nicht möglich, denn hierbei dürften keine Resource-Dateien Verwendung finden. Speicher, den sich ein Accessory reserviert, wird von der aktuellen Hauptapplikation genommen. Da das Desktop unter *GEM 2.x* als ganz normales Programm benutzt wird, würde sich Wodan den Speicher vom Desktop nehmen. Wird das Desktop verlassen und ein anderes Programm nachgeladen, muß auch der Speicher freigegeben werden. Folglich müßte die Resource-Datei bei nächster Gelegenheit erneut geladen werden, was logischerweise ein sehr unschöner Effekt ist. Abhilfe wäre zu erreichen, wenn statt der Resource-Datei alle Deklarationen der Dialogboxen vom Programm aus vorgenommen würden. Leider existiert zur Zeit noch kein Programm, welches dem Programmierer/der Programmiererin die recht aufwendige Arbeit abnimmt und Resource-Dateien direkt in Modula-2-Quelltext umwandelt.

Beginnen wir mit der Codierung des Programms, womit nicht die Erstellung des Maschinen-Codes gemeint ist, sondern der Aufbau der Listings. Innerhalb der vier Folgen soll versucht werden, die Top-Down-Methode einzuhalten. Wir beginnen heute mit dem Hauptmodul und werden schrittweise feiner unterteilen.

Das Hauptmodul beinhaltet noch nichts, was mit der GEM-Umgebung zu tun hat. Es wird lediglich das Programm installiert und je nach Parametereinstellung bzw. Reaktion der Funktion *HandleAcc* ein Vektor verbogen - auf eine neue Routine gelenkt - bzw. wieder ausgehängt. Alles weitere soll uns momentan nicht interessieren.

Den *IKBD*-Systemvektor biegen wir um auf die Routine *hdlKbd*, die ihrerseits *kbdWriteString* aufruft. Diese beiden Routinen werden vom Modula-2-Programm selbst nicht aufgerufen. Von *hdlKbd* werden die in den Tastatur-Puffer gestellten Tasten abgefangen; d.h. bevor ein anderes Programm sich die Tastendrücke aus dem Puffer holt, schaut erst einmal *hdlKbd* nach, ob denn nichts zu ersetzen ist. Steht eine Taste im Puffer, die in Verbindung mit Alternate gedrückt wurde, wird dieser Tastendruck durch eine komplette Zeichenkette ersetzt. *IsAlternate* gibt nur dann *TRUE* zurück, wenn Alternate in Verbindung mit A bis Z bzw. 0 bis 9 gedrückt wurde, also genau das, was wir benötigen.

Die Routine *Aktion* erzeugt beim ersten Aufruf den Entry-Code für die Modula-2-Routine. Danach wird der Vektor installiert, sofern das noch nicht geschehen ist. Im Falle, daß die eigene Routine herausgenommen werden soll, wird die Installation einfach entfernt.

Die Initialisierungsroutine initialisiert das Accessory und teilt einem untergeordneten Modul die Adresse des Arrays mit den Zeichenketten mit. Das Hauptprogramm selbst wartet in einer Endlosschleife ab, ob Wodan installiert werden soll oder nicht.

Die vier Definitionsmodule *CommunicationWodan*, *GEMUtility*, *HandleWodan* und *XBRA* sollen an dieser Stelle schon zeigen, welche Routinen noch zu Wodan gehören. Für das Hauptmodul werden Routinen aus *HandleWodan* und *XBRA* benötigt, die anderen beiden stehen für *HandleWodan* zur Verfügung. Erklärungen zu diesen Modulen folgen in den drei nächsten Folgen.

Wenn Sie nun schon beginnen möchten, die Listings einzugeben, können Sie die Module auch bereits compilieren. Es muß dabei darauf geachtet werden, daß die Definitionsmodule vor dem Hauptmodul übersetzt sein müssen, da der Compiler sonst die Funktionsdeklarationen nicht finden kann.

Zum Schluß noch ein Wort zum Tastatur-Puffer. In ihn kann auch ohne die Funktionen der Megamax-Modula-2-Modulen geschrieben werden. Den Schlüssel hierzu liefert die *XBIOS*-Funktion 14 - *Iorec* -, die einen Zeiger unter anderem auf den Tastatur-Puffer zurückgibt. Somit dürfte sich Wodan oder eine ähnliche Application auch mit anderen Entwicklungssystemen wie C oder Pascal realisieren lassen.

Bis zur nächsten Folge...

Dietmar Rabich

Literatur:

[1] Atari ST Profibuch, H.-D. Jankowski/ J. F. Reschke/ D. Rabich, 6. Auflage, Sybex-Verlag 1988/89, S. 86, 107, 915.

[2] dtv-Lexikon, dtv 1980.

```
 1:  (***********************************************)
 2:  (* Modulname             : WodanAcc            *)
 3:  (* Copyright             : MAXON Computer GmbH *)
 4:  (* Datum                 :  1. Juni 1989       *)
 5:  (* letztes Edier-Datum   :  1. September 1989  *)
 6:  (* Version               : 1.00a               *)
 7:  (* Entwicklungssystem    : Megamax Modula-2    *)
 8:  (***********************************************)
 9:
10:  MODULE WodanAcc;
11:  (*$Q+,M-,N-,V+,P-,R-,S-*)
12:
13:  (* String-Routinen *)
14:  FROM Strings         IMPORT String, Length, Empty;
15:
16:  (* TOS-Routinen *)
17:  FROM XBIOS           IMPORT SuperExec,
                                KeyboardVectors;
18:
19:  (* verschiedene, systemnahe Routinen *)
20:  FROM MOSGlobals      IMPORT MemArea;
21:  FROM Calls           IMPORT SysNewCaller,
                                CallExtRegs, Registers;
22:  FROM SysUtil1        IMPORT SuperPeek, SuperLPeek,
                                SuperLPoke;
23:  FROM SYSTEM          IMPORT ADR, VAL, ADDRESS,
                                LONGWORD, WORD, BYTE;
24:  FROM XBRA            IMPORT Carrier, Install,
                                Installed, Create,
                                Remove,
25:                             PreviousEntry;
26:
27:  (* Keyboard-Routinen *)
28:  FROM KbdCtrl         IMPORT LookKey, GetKey, Key,
                                PutKey,CtrlKey, CtrlSet;
29:  FROM Keyboard        IMPORT KeyCap, IsAlternate,
                                SpecialKey, SpecialCode;
```

→

```
30:
31:  (* Routinen für die GEM-Umgebung *)
32:  FROM HandleWodan  IMPORT StringArray,
                             InitParameter,
                             HandleAcc, CancelAcc,
33:                          InitAcc, NoInstallation,
                             SetStringAdr;
34:
35:
36:  CONST MyXBRA   = 'DRRH'; (* reservierte XBRA-
                               Kennung! *)
37:  (* Alle mit DRR beginnenden XBRA-Kennungen sind
       für      *)
38:  (* Dietmar (R.) Rabich reserviert. H steht für
       Hermes.   *)
39:  (* Der Name von Wodan war während der Testphase
       Hermes.   *)
40:
41:
42:  VAR   at, entry, vector,
43:        kbdentry             : ADDRESS;
44:        kbdstack             : ARRAY [1..800] OF
                                  CARDINAL;
45:        kbdwsp, termwsp      : MemArea;
46:        kbdv                 : POINTER TO ADDRESS;
47:        carrier              : Carrier;
48:        ok, InstOK           : BOOLEAN;
49:        macro                : StringArray;
50:
51:
52:  (* String in den Tastaturpuffer schreiben *)
53:  PROCEDURE kbdWriteString (s : ARRAY OF CHAR);
54:
55:    VAR i : CARDINAL;
56:        k : Key;
57:
58:    BEGIN
59:      IF ~Empty (s)
60:      THEN
61:        k.scan := 0;            (* kein Scan-Code    *)
62:        k.ctrl := CtrlSet{};  (* keine Sondertaste*)
63:        FOR i := 0 TO Length (s) - 1 (* Character
                                        für Character
                                        in den *)
64:        DO                        (* Puffer
                                        schreiben *)
65:          k.ch := s[i];
66:          PutKey (k, ok)
67:        END
68:      END
69:    END kbdWriteString;
70:
71:
72:  (* nachschauen, ob Taste bereitsteht und ggf.
       String schreiben *)
73:  PROCEDURE hdlKbd (VAR r : Registers);
74:
75:    VAR k : Key;
76:
77:    BEGIN
78:      CallExtRegs (PreviousEntry (entry), r);
79:      LookKey (k, ok);      (* Taste im
                                  Tastaturpuffer?  *)
80:      IF ok
81:      THEN
82:        IF IsAlternate(k)   (* mit Alternate
                                  gedrückt?        *)
83:        THEN
84:          GetKey (k, ok);   (* Taste aus dem
                                  Puffer entfernen *)
85:          (* dafür unseren String hineinschreiben
             ...                                   *)
86:          kbdWriteString (macro[SpecialKey(k)])
87:        END
88:      END
89:    END hdlKbd;
90:
91:
92:  (* Wodan installieren bzw. entfernen *)
93:  PROCEDURE Aktion (On : BOOLEAN);
94:
95:    BEGIN
96:
97:      IF On
98:      THEN
99:        IF ~InstOK (* schon einmal installiert? *)
100:       THEN
101:         kbdwsp.bottom := ADR (kbdstack);
                        (* Stackbereich bestimmen    *)
102:         kbdwsp.length := SIZE (kbdstack);
103:         SysNewCaller (hdlKbd, FALSE, kbdwsp,
              kbdentry); (* Entry-Code
                            erzeugen*)
104:         InstOK := kbdentry # NIL
105:       END;
106:       IF InstOK
107:       THEN
108:         vector := ADDRESS (KeyboardVectors ()) +
                       $20L; (* Vektor holen *)
109:         IF NOT Installed (MyXBRA, vector, at)
110:         THEN
111:           Create (carrier, MyXBRA, kbdentry,
                       entry);    (* installieren *)
112:           Install (entry, at)
113:         END
114:       ELSE
115:         NoInstallation
116:       END
117:     ELSE
118:       vector := ADDRESS (KeyboardVectors()) +
                     $20L;  (* Vektor holen     *)
119:       IF Installed (MyXBRA, vector, at)
                          (* installiert?     *)
120:       THEN
121:         Remove(at)
                          (* dann entfernen *)
122:       END
123:     END
124:   END Aktion;
125:
126:
127: (* Wodan installieren *)
128: PROCEDURE Init () : BOOLEAN;
129:
130:   VAR success : BOOLEAN;
131:
132:   BEGIN
133:
134:    InstOK := FALSE;
135:
136:    success := InitAcc ();   (* Accessory
                                   installieren *)
137:
138:    SetStringAdr (ADR(macro)); (* Adresse der
                                     Strings
                                     bekanntgeben *)
139:
140:    IF success
141:    THEN
142:      Aktion (InitParameter()) (* erste
                                     Installation *)
143:    END;
144:
145:    RETURN success
146:
147:   END Init;
148:
149:
150: (* Hauptprogramm *)
151: BEGIN
152:  IF Init() THEN              (* installieren    *)
153:
154:   WHILE TRUE DO              (* Endlosschleife *)
155:    Aktion (HandleAcc())
156:   END
157:
158:  ELSE
159:
160:   CancelAcc
161:
162:  END
163: END WodanAcc.
```

→

```
 1: (************************************************)
 2: (* Modulname             : CommunicationWodan
                             (DEFINITION)          *)
 3: (* Copyright             : MAXON Computer GmbH  *)
 4: (* Datum                 : 10. Juni 1989        *)
 5: (* letztes Edier-Datum: 1. September 1989       *)
 6: (* Version               : 1.00a                *)
 7: (* Entwicklungssystem : Megamax Modula-2        *)
 8: (************************************************)
 9:
10: DEFINITION MODULE CommunicationWodan;
11: (*$Q+,M-,N-,V+,P-,R-,S-*)
12:
13:
14: FROM AESEvents    IMPORT MessageBuffer;
15: FROM SYSTEM       IMPORT ADDRESS;
16:
17:
18: (* Konstanten für Message-Type *)
19: CONST ForeignCall   = $4001;          (*
        Accessory-Fremdaufruf *)
20:       ForeignAnswer = $4002;
21:
22:
23: (* Sendet an aufrufende Applikation Message
      zurück *)
24: PROCEDURE SendToAppl (msgBuffer : MessageBuffer;
            lParasAdr : ADDRESS);
25:
26:
27: END CommunicationWodan.
```

```
 1: (************************************************)
 2: (* Modulname             : HandleWodan(DEFINITION)*)
 3: (* Copyright             : MAXON Computer GmbH  *)
 4: (* Datum                 :  1. Juni 1989        *)
 5: (* letztes Edier-Datum: 11. Oktober 1989        *)
 6: (* Version               : 1.00a                *)
 7: (* Entwicklungssystem : Megamax Modula-2        *)
 8: (************************************************)
 9:
10: DEFINITION MODULE HandleWodan;
11: (*$Q+,M-,N-,V+,P-,R-,S-*)
12:
13:
14: (* Importe *)
15: FROM GEMGlobals IMPORT PtrObjTree;
16: FROM Keyboard   IMPORT SpecialCode;
17: FROM Strings    IMPORT String;
18: FROM SYSTEM     IMPORT ADDRESS;
19:
20:
21: (* Typen *)
22: TYPE StringArray      = ARRAY [alt1..altZ] OF
                            String;
23:      LokalParameter   = RECORD (* für Fremdaufruf
                            und internen Gebrauch *)
24:                           Aktiv     : BOOLEAN;
25:                           Position  :
                                        SpecialCode;
26:                           StringAdr : POINTER
                                      TO StringArray
27:                         END;
28:
29:
30: (* Fehlermeldung, falls Installation nicht
      möglich *)
31: PROCEDURE NoInstallation;
32:
33: (* Setzt die Adresse der Strings *)
34: PROCEDURE SetStringAdr (SAdr : ADDRESS);
35:
36: (* Accessory initialisieren,                   *)
37: (* Rückgabe: TRUE, falls erfolgreich, FALSE
      sonst                     *)
38: PROCEDURE InitAcc : BOOLEAN;
39:
40: (* Parameter initialisieren,                   *)
41: (* Rückgabe: TRUE, falls Wodan aktiv, FALSE,
      falls Wodan passiv *)
42: PROCEDURE InitParameter : BOOLEAN;
43:
44: (* Accessory-Ablauf,                           *)
45: (* Rückgabe: TRUE, falls Wodan aktiv, FALSE,
      falls Wodan passiv *)
46: PROCEDURE HandleAcc : BOOLEAN;
47:
48: (* Accessory canceln (Warteschleife) *)
49: PROCEDURE CancelAcc;
50:
51:
52: END HandleWodan.
```

```
 1: (************************************************)
 2: (* Modulname            : GEMUtility(DEFINITION)*)
 3: (* Copyright            : MAXON Computer GmbH   *)
 4: (* Datum                :  3. Juni 1989         *)
 5: (* letztes Edier-Datum : 10. Juni 1989          *)
 6: (* Version              : 1.00a                 *)
 7: (* Entwicklungssystem   : Megamax Modula-2      *)
 8: (************************************************)
 9:
10: DEFINITION MODULE GEMUtility;
11: (*$Q+,M-,N-,V+,P-,R-,S-*)
12:
13:
14: (* Importe *)
15: FROM GEMGlobals IMPORT PtrObjTree, ObjState;
16: FROM GrafBase   IMPORT Rectangle;
17:
18:
19: (* Maus als Biene, Sanduhr, ... *)
20: PROCEDURE ShowBusy;
21:
22: (* Maus als Pfeil *)
23: PROCEDURE ShowArrow;
24:
25: (* Maus zeigen *)
26: PROCEDURE ShowMouse;
27:
28: (* Maus verstecken *)
29: PROCEDURE HideMouse;
30:
31: (* Platz, den das Objekt belegt *)
32: PROCEDURE objectSpace (obj : CARDINAL) :
              Rectangle;
33:
34: (* Platz, den das Objekt bzgl. Offset belegt *)
35: PROCEDURE objOffsetSpace (obj : CARDINAL) :
              Rectangle;
36:
37: (* Objekt-Status entfernen *)
38: PROCEDURE clearObjState (obj : CARDINAL; which :
              ObjState; redraw : BOOLEAN);
39:
40: (* Objekt-Status setzen *)
41: PROCEDURE setObjState (obj : CARDINAL; which :
              ObjState; redraw : BOOLEAN);
42:
43: (* Zeichenkette holen *)
44: PROCEDURE getTextString (tree    : PtrObjTree;
45:                          obj     : CARDINAL;
46:                          VAR str : ARRAY OF
                                       CHAR);
47:
48: (* Zeichenkette setzen *)
49: PROCEDURE setTextString (tree    : PtrObjTree;
50:                          obj     : CARDINAL;
51:                          VAR str : ARRAY OF
                                       CHAR);
52: (* VAR nur wegen der Geschwindigkeit *)
53:
54: (* Character setzen *)
55: PROCEDURE setTextChar (tree   : PtrObjTree;
56:                        obj    : CARDINAL;
57:                        char   : CHAR);
58:
59: (* Dialog vorbereiten *)
60: PROCEDURE prepareBox (tree : PtrObjTree) :
              Rectangle;
61:
62: (* Dialog nachbereiten *)
63: PROCEDURE releaseBox (tree : PtrObjTree; space :
              Rectangle);
64:
65:
66: END GEMUtility.
```

→

```
  1:  DEFINITION MODULE XBRA;
  2:
  3:  (*
  4:   * Universelle XBRA-Funktionen
       Version 1.1  vom  18.6.89
  5:   *
  6:   * Erstellt von Thomas Tempelmann.
  7:   *
  8:   * Die vorliegende Version ist unter dem Megamax
       Modula-2-System compilierbar,
  9:   * für Hänisch- und SPC-Modula sind nur wenige
       Anpassungen nötig.
 10:   *
 11:   * Die vorhandenen Funktionen bieten alles, um
       auf einfache Weise installierte
 12:   * Vektoren zu erkunden (wahlweise alle oder
       einen spezifischen) und sie
 13:   * korrekt ein-, bzw. wieder auszutragen.
 14:   *
 15:   * Wurde eine Funktion installiert, kann durch
       die Funktion 'PreviousEntry' die
 16:   * vorher installierte Routine ermittelt werden.
       Damit ist es dann möglich,
 17:   * den Vorgänger in der Funktion selbst
       aufzurufen, falls dies nötig wäre.
 18:   * Die zum Aufrufen notwendigen Funktionen sind
       jedoch nicht Bestandteil
 19:   * dieser XBRA-Library, da sie erstens Compiler-
       spezifisch und zweitens
 20:   * sie je nach Anwendung sehr unterschiedlich
       implementiert werden müssen.
 21:   *
 22:   * Die Funktionen sind so ausgelegt, daP sie
       normalerweise im User-Mode
 23:   * aufgerufen werden und selbst beim Zugriff
       über die Vektoren in den Super-
 24:   * visor-Mode wechseln (das Programm darf sich
       aber auch bereits im Supervi-
 25:   * sor-Mode befinden). Dies erleichtert dem
       Programmierer die Anwendung der
 26:   * Funktionen. Da die Anwendungen beim
       Installieren von Vektoren in der Regel
 27:   * nicht zeitkritisch ausgelegt sein brauchen,
       sollte man diese Komfortabilität
 28:   * dem leichten Zeitverlust durch - meist - zwei
       statt nur einem Wechsel vom
 29:   * User- in den Supervisor-Mode vorziehen.
 30:   *
 31:   *
 32:   * Hier noch eine allgemeine Beschreibung zur
       Anwendung der Funktionen:
 33:   *
 34:   * Folgendermaßen sieht eine XBRA-Installation
       aus:
 35:   *
 36:   *   vector:= 400H; (* z.B. der 'etv_timer'-
                          Vektor *)
 37:   *   IF NOT Installed ('Test', vector, at) THEN
 38:   *     Create (carrier, 'Test', ADDRESS
       (TestProzedur), entry);
 39:   *     Install (entry, at)
 40:   *   END;
 41:   *
 42:   * 'Installed' prüft, ob die Funktion schon mit
       XBRA-Kennung installiert
 43:   * ist. Wenn nicht, wird mit 'Create' ein XBRA-
       Header erzeugt, der neben
 44:   * der XBRA-Informationen auch eine
       Sprunganweisung enthält. Der so
 45:   * erzeugte Header wird dann mit 'Install' als
       erster neuer Vektor
 46:   * eingetragen und die XBRA-Verkettung erzeugt.
 47:   *
 48:   * Da je nach Implementation verschiedene
       Prozeduren mit beliebigen
 49:   * Parametern verwendet werden könnten, und der
       XBRA-Header nur einfach
 50:   * dazwischengesetzt wird, ist die
       Prozeduradresse 'call' als ADDRESS
 51:   * deklariert. Für die korrekte
       Parameterübergabe haben nicht die XBRA-
 52:   * Funktionen zu sorgen, sondern schon die
       Routine, die zu installierende
 53:   * Funktion über den Vektor aufruft.
 54:   *
 55:   * Zum Vergleich obige Installation ohne XBRA:
 56:   *   vector:= 400H;  (* VAR vector: POINTER TO
                            ADDRESS *)
 57:   *   vector^:= ADDRESS (TestProzedur);
 58:   * Falls es Probleme gibt, sollte erst das
       Programm ohne XBRA zum Laufen
 59:   * gebracht werden, und dann erst die XBRA-
       Installation eingefügt werden.
 60:   * Z.B. ist beim Megamax-System zu beachten, daß
       normalerweise Installationen
 61:   * über externe Vektoren über die Funktionen aus
       dem Modul 'Calls' vorgenommen
 62:   * werden sollten. Dies bleibt auch so, wenn
       dann die XBRA-Funktionen zuhilfe
 63:   * genommen werden!
 64:   *
 65:   * Soll die Funktion später wieder aus der
       Vektor-Kette ausgehängt werden,
 66:   * geht das so:
 67:   *
 68:   *   IF Installed ('Test', vector, at) THEN
 69:   *     Remove (at)
 70:   *   END
 71:   *
 72:   * Bei 'PreviousEntry' (s.o.) wird der bei
       'Create' erhaltene 'entry'-Wert
 73:   * wieder übergeben, um z.B. in 'TestProzedur'
       den Vorgänger zu
 74:   * ermitteln und dann ggf. aufzurufen.
 75:   *
 76:   * 'Query' dient dazu, alle installierten XBRA-
       Kennungen einer Vektor-
 77:   * Kette zu ermitteln. Dabei können auch z.B.
       mit folgender Routine alle
 78:   * installierten XBRA-Vektoren ausgehängt werden:
 79:   *
 80:   *   PROCEDURE step (at: ADDRESS; name: ID):
                 BOOLEAN;
 81:   *     BEGIN
 82:   *       (*
 83:   *        * Hier könnten der jeweilige Vorgänger
                 mit
 84:   *        *  'PreviousEntry ( Entry (at) )'
 85:   *        *  oder die Adr. der aufgerufenen
                 Prozedur mit
 86:   *        *  'Called (at)'  ermittelt und
                 angezeigt werden.
 87:   *        *)
 88:   *       IF prev # NIL THEN (* ist dies ein XBRA-
                                 Eintrag? *)
 89:   *         Remove (at)      (* -> nur dann kann
                                 er entfernt werden *)
 90:   *       END;
 91:   *       RETURN TRUE        (* weitermachen,
                                  solange die Kette
                                  weitergeht *)
 92:   *     END step;
 93:   *
 94:   *   PROCEDURE RemoveAll (vector: ADDRESS);
 95:   *     BEGIN Query (vector, step) END RemoveAll;
 96:   *
 97:   *)
 98:
 99:  FROM SYSTEM IMPORT ADDRESS;
100:
101:  TYPE
102:
103:    ID = ARRAY [0..3] OF CHAR;
           (* String zur Aufnahme der Kennung *)
104:
105:    JmpCarrier = RECORD
          (* Interne Datenstruktur!        *)
106:                 jmpInstr: CARDINAL;
          (* -  nicht darauf zugreifen! *)
107:                 operand: ADDRESS
108:               END;
109:
110:    Carrier = RECORD       (* Interne Datenstruktur
                        - nicht darauf zugreifen! *)
111:              magic: ID;       (* CONST 'XBRA' *)
112:              name : ID;       (* individuelle
                                      Kennung *)
113:              prev : ADDRESS; (* voriger
                                      Vektor *)
114:              entry: JmpCarrier;
115:            END;
```

→

```
116:
117:   QueryProc = PROCEDURE ( (* at   : *) ADDRESS,
118:                           (* name: *) ID      ):
                               (* continue:*)
                               BOOLEAN;
119:
120:
121:  (*
122:   * Funktionen für die XBRA-Installation
123:   *
124:   *)
125:
126:  PROCEDURE Create (VAR use: Carrier; name: ID;
                 call: ADDRESS;
127:                        VAR entry: ADDRESS);
128:    (*
129:     * Erzeugt einen XBRA-Header mit einer
         Sprunganweisung zur Prozedur 'call'.
130:     * Achtung: die Carrier-Variable muß global
         (statisch) deklariert sein -
131:     * sie muß so lange erhalten bleiben, wie
         die XBRA-Einbindung besteht!
132:     * Der erhaltene 'entry'-Wert kann daraufhin
         mittels der Prozedur 'Install'
133:     * in den gewünschten Vektor eingetragen
         werden.
134:     *)
135:
136:  PROCEDURE Installed (name: ID; vector: ADDRESS;
                 VAR at: ADDRESS): BOOLEAN;
137:    (*
138:     * Wird 'name' in Kette ab 'vector' gefunden,
         enthält 'at' die Adresse
139:     * des Vektors auf den Funktionseinsprung
         (welcher Teil von 'Carrier' ist).
140:     * Wird 'name' nicht gefunden, ist 'at'=vector
141:     *)
142:
143:  PROCEDURE Install (entry: ADDRESS; at: ADDRESS);
144:    (*
145:     * Fügt einen XBRA-Header 'entry' im Vektor
         'at' ein. Der alte Vektorinhalt
146:     * wird im XBRA-Header gesichert und kann
         mittels 'PreviousEntry' abgefragt
147:     * werden.
148:     *)
149:
150:  PROCEDURE Remove (at: ADDRESS);
151:    (*
152:     * Fügt den XBRA-Header, auf den der Vektor
         bei 'at' zeigt, aus.
153:     * In den Vektor wird wieder der Vorgänger
         eingetragen.
154:     *)
155:
156:  (*
157:   * Funktionen zum Abfragen XBRA-Informationen
158:   *
159:   *)
160:
161:  PROCEDURE Query (vector: ADDRESS; with:
                 QueryProc);
162:    (*
163:     * Ruft 'with' für alle im Vektor 'vector'
         installierten Funktionen auf,
164:     * solange sie durch XBRA-Strukturen verbunden
         sind.
165:     * Die 'with'-Funktion kann 'FALSE'
         zurückgeben, um die Aufrufe vorzeitig
166:     * zu beenden.
167:     *)
168:
169:  PROCEDURE Entry (at: ADDRESS): ADDRESS;
170:    (*
171:     * Liefert die Adresse, auf die der Vektor
         'at' zeigt.
172:     * Dies ist der "Entry", vor dem ggf. die XBRA-
         Struktur steht.
173:     * Das Ergebnis dieser Funktion kann z.B. für
         die 'PreviousEntry'-Funktion
174:     * verwendet werden, sollte jedoch nicht zur
         rein informativen Ermitttelung
175:     * der in 'at' installierten Funktion
         verwendet werden - dafür ist 'Called'
176:     * (s.u.) vorgesehen!
177:     *)
178:
179:  PROCEDURE PreviousEntry (entry: ADDRESS):
                 ADDRESS;
180:    (*
181:     * Liefert den "Entry", der vor dem
         angegebenen "Entry" installiert ist.
182:     * Diese Funktion kann z.B. in der eigenen
         installierten Funktion dazu
183:     * benutzt werden, den Vorgänger aufzurufen
         (hier sollte aus Effizienz-
184:     * gründen nicht 'Called' verwendet werden),
         um eine Aufrufkette zu reali-
185:     * sieren (ist z.B. beim 200Hz-Vektor sinnvoll,
         da hier mehrere Routinen
186:     * hintereinander installiert werden, die alle
         sich nacheinander aufrufen).
187:     * Zu diesem Zweck muß dann der bei 'Create'
         erhaltene 'entry'-Wert über-
188:     * geben werden. Achtung: Da damit gerechnet
         werden muß, daß während der
189:     * Lebzeit des Programms im Speicher die XBRA-
         Verkettung jederzeit ver-
190:     * ändert werden kann, darf nicht einmalig
         fest die Vorgängeradr. abgefragt
191:     * und dann über eine Programmlokale Variable
         adressiert werden, sondern
192:     * muß immer genau dann, wenn sie benötigt
         wird, mit dieser Funktion ab-
193:     * gefragt werden. Es sind dabei keine
         nennenswerten Zeitverluste zu be-
194:     * fürchten.
195:     * Wenn 'entry'=NIL, oder kein Vorgänger
         ermittelt werden kann (keine XBRA-
196:     * Struktur vorhanden), wird NIL geliefert.
197:     *)
198:
199:  PROCEDURE Called (at: ADDRESS): ADDRESS;
200:    (*
201:     * Liefert die korrekte
         Programmeinsprungstelle, die über den
         Vektor 'at'
202:     * erreicht wird. Dies wäre normalerweise
         identisch mit dem Ergebnis der
203:     * 'Entry'-Funktion, jedoch wird hier erkannt,
         wenn diese "Entry"-Stelle
204:     * nur die von diesem Modul erzeugte
         Sprunganweisung in die eigentliche
205:     * Funktion ist, die bei der 'Install'-
         Funktion angegeben wurde. Dann
206:     * wird jene Funktionsadresse geliefert.
207:     *)
208:
209:  END XBRA.
```



XModem und DATEX-P

Nachdem wir nun längere Zeit keine DFÜ-Ecke abdruckt haben, geht es diesmal mit umso mehr Elan weiter. Damit es auch richtig interessant wird, verlosen wir in dieser Ausgabe 5 Interlink ST-Terminal-Programme, die freundlicherweise von BELA in Eschborn gespendet wurden.

Interlink ST hat zum Beispiel eine komplette VT100-Emulation mit IBM-Grafikzeichen, eine eigene kleine Mailbox-Funktion, Autodial und vieles mehr. Wenn Sie eines der 5 Programme gewinnen möchten, müssen Sie uns lediglich eine Postkarte schicken, auf die Sie die Namen von 3 Mailbox-Netzen schreiben. Kleiner Tip: Schauen Sie mal in Ausgabe 10 der ST-COMPUTER. Der Einsendeschluß für das Preisausschreiben ist Freitag, der 30.1.1990. Der Rechtsweg ist ausgeschlossen. Schicken Sie Ihre Postkarte an MAXON Computer GmbH, Stichwort: Interlink, Industriestraße 26, 6236 Eschborn.

Grundsätzliches

Einige Briefe haben mich erreicht, in denen noch viele grundsätzliche Fragen zur DFÜ gestellt wurden. Deshalb hier noch eine kurze Einführung, wie Sie sich zum Beispiel in eine Mailbox einloggen können. Nehmen Sie sich Ihren Akustikkoppler oder Ihr Modem und schließen Sie es mit einem RS232-Kabel an die Schnittstelle des STs an. Ein Modem müssen Sie, sofern das die Bundespost noch nicht für Sie erledigt hat, noch an die Telefonleitung anschließen. Nun nehmen Sie die Nummer einer Mailbox und wählen Sie. Wenn Sie einen Koppler besitzen, wählen Sie sie per Hand mit dem Telefon, bei einem Hayes-kompatiblen Modem geben Sie [beispielsweise bei der Nummer (069) 683584] das Kommando "AT DP 069683584" ein, bei einem postzugelassenen Hayes-kompatiblen Modem heißt das Kommando "AT DP 069 683584". Beim Modem müssen Sie jetzt nur noch warten, bis die Verbindung zustandegekommen ist. Bei einem Koppler müssen Sie warten, bis Sie einen Carrier hören. Drücken Sie dann den Hörer in den Koppler und betätigen Sie einige Male Return, bis die ersten Zeichen auf dem Bildschirm erscheinen. Sollten Probleme auftauchen, sehen Sie bitte in den Tabellen 1 und 2 nach, wie Sie sie beheben können.

Los geht's

Nachdem Sie Return gedrückt haben, werden Sie aufgefordert, sich mit einem Usernamen einzuloggen. Wenn Sie vorher noch nicht in der Mailbox angerufen haben, geben Sie als Namen "GAST" ein. Sollte Sie die Mailbox nun trotzdem noch nach einem Paßwort fragen, geben Sie auch hier "GAST" ein. Werden Sie als

Problem	Abhilfe
Es kommt nur Datenmüll.	Haben Sie die Parameter richtig eingestellt? Die meisten Mailboxen benutzen die Parameter 8 Daten-Bits, keine Parität, 1 Stop-Bit oder 7 Daten-Bits, gerade Parität, 1 Stop-Bit.
Die Mailbox hebt zwar ab, aber das Mailbox-Modem erkennt meinen Carrier nicht.	Sie versuchen, in eine Mailbox zu kommen, die im Bell-Modus arbeitet oder ein Discovery-Modem benutzt. Schalten Sie den Koppler auf den Bell-Modus. Bringt dies nichts, schalten Sie den Koppler erst an, wenn der dritte Ton des Modems kommt!
Ich kann alles erkennen, die Box erkennt mich nicht.	Verfahren Sie wie beim Punkt "Es kommt nur Datenmüll".
Ich besitze zwar einen 2400-Baud-Koppler, kann aber kaum 2400-Baud-Boxen anrufen!	Besitzen Sie einen Dataphon 2400 B-Koppler, dann stoppen Sie die Versuche. Dieses Gerät erkennt keine Discovery- und BEST-Modems. Rufen Sie statt dessen mit 300 Baud an.

Tabelle 1: Probleme und deren Lösungen bei Akustikkopplern

Problem	Abhilfe
Es kommt nur Datenmüll.	Haben Sie die Parameter richtig eingestellt? Die meisten Mailboxen benutzen die Parameter 8 Daten-Bits, keine Parität, 1 Stop-Bit oder 7 Daten-Bits, gerade Parität, 1 Stop-Bit.
Immer wenn es klingelt, erkennt mein Modem BUSY.	Geben Sie nach dem Wählkommando einige Kommazeichen an.
Kurz bevor der Carrier kommt, legt mein Modem auf.	Setzen Sie mit dem Kommando "AT S9=60 S7=60" die Carrier-Wartezeit etwas höher.
Während mein Modem wählt, erkennt es BUSY.	Geben Sie nach dem Wählkommando einige Kommazeichen an.
Das Modem CONNECTet nicht.	Das Modem ist im BELL-Modus aktiviert. Geben Sie das Kommando "AT B0" ein.

Tabelle 2: Probleme und deren Lösungen bei Modems

GAST nicht akzeptiert, geben Sie "GUEST" ein.

Nun sollten Sie im System eingeloggt sein. Geben Sie den Befehl "HILFE" oder "HELP" ein, um einen Hilfstext zu bekommen. Diesen drucken Sie sich am besten aus, um ihn in aller Ruhe zu Hause lesen zu können. So können Sie bei Ihrem nächsten Anruf schon wesentlich besser mit der Mailbox umgehen.

DATEX-P

Wenn Sie das Hobby DFÜ intensiver betreiben, werden Sie spätestens bei der nächsten Telefonrechnung feststellen, daß Joggen ein billigeres Hobby ist. So kostet die Verbindung Deutschland - USA zirka DM 4,60 pro Minute, selbst innerhalb der Bundesrepublik sollte man sich bei telefonischen Ausflügen auf Mailboxen der näheren Umgebung beschränken. Nun brauchen Sie allerdings nicht darauf zu warten, daß Ihre Erbtante zu den Vorfahren geht, bevor Sie sich wieder eine Computerverbindung in die USA erlauben können. Alternativ zum Telefon gibt es nämlich noch eine Möglichkeit, Ihren elektronischen Hausfreund mit der Welt kommunizieren zu lassen - den Datenvermittlungsdienst der Deutschen Bundespost, kurz DATEX. In Großstädten, z.B. in Frankfurt, genügt es, den PAD (Packet Assembly/Disassembly Facility - Anpassungseinrichtung zum DATEX-P-Netz) mit 20281 (300 Baud) oder 20291 (1200 Baud) (Nummern aus Frankfurt) anzuwählen, sich mit einer NUI (Network User Identification) sowie einem Paßwort zu identifizieren, und schon kann die Reise per Computer beginnen. Im Gegensatz zum Telefon wird bei DATEX-P nicht nur die Zeit als Berechnungsgrundlage genommen; auch die Anzahl der übertragenen Datenpakete sowie deren Umformung müssen Sie bezahlen. Das klingt zunächst schlimmer, als es ist. Nehmen wir als Beispiel ein Textfile von 12000 Bytes, welches wir mit 1200 Baud in die USA übertragen wollen. Dazu würden wir, optimale Leitungsverbindung vorausgesetzt, zirka 100 Sekunden benötigen. Also würden an Telefongebühren zirka 7,70 DM fällig werden. Die DATEX-P-Rechnung sähe wie folgt aus:

Anruf zum PAD (Ortstarif):	DM 0,23	
Zugangsgebühr (1200 Baud):	DM 0,05	
Zeitgebühr (100 Sekunden):	DM 0,35	(DM 0,15/Minute)
Segmentgebühren (für 157 Segmente):	DM 1,40	(9 Pfennig/Segment)
Anpassungsgebühren 100 Sekunden:	DM 0,09	(DM 0,06/Minute)
	=======	
Kosten des File-Transfers in die USA:	DM 2,12	

Wie aber kommt man nun in die Reihen der gebührensparenden DATEX-P-Nutzer? Einfach eine NUI beantragen. Am besten im Telefonladen oder beim Postamt. Mit 15 DM Grundgebühr im Monat ist man dabei. Allerdings sollten Sie sich (ebenfalls bei der Post) vorher erkundigen, wo in Ihrer Nähe der Einwählpunkt zum DATEX-P-Netz zu finden ist. Was nützt die billigste DATEX-P-Verbindung, wenn Ihr PAD nur zu Ferngesprächsgebühren erreichbar ist? Über eines sollten Sie sich allerdings im klaren sein: Über das DATEX-P-Netz erreichen Sie nur Mailbox-Systeme, die auch an dieses Netzwerk angeschlossen sind, zum Beispiel (siehe untenstehende Tabelle).

NUA	Name	Systeminfo	Paßwort
0270448112	ECHO	Datenbank der EG	TRAIND
45711040009	GTC	kommerzielle Mailbox	GAST
45511090835	cosmonet	kommerzielle Mailbox	GAST
45890040004	ALTOS	Multiuser-Dialog	GUEST

Jetzt wissen Sie, wie Sie in Zukunft Ihre Telefonrechnung in Grenzen halten können. Doch Vorsicht: Auch DATEX-P-Verbindungen können teuer werden. Und falls nun auch Ihre DATEX-Rechnungen existenzgefährdende Höhen erreichen, sollten Sie vielleicht doch das Hobby wechseln.

Wo denn?

Wo gibt's Mailboxen? Es dürfte nicht schwer sein, auch in Ihrem Nahbereich eine Mailbox zu finden. Wenn Sie schon Interesse haben, können Sie eine der folgenden Nummern anrufen. In diesen Mailboxen können Sie sich im Brett "Mailbox-Listen" die jeweils aktuelle Mailbox-Liste ansehen und finden dann schnell eine Mailbox in Ihrem Nahbereich. PEC: 069/683584; PCB: 0511/6040070; Paderbox: 05251/21286; Asylum: 02372/13383.

Das XModem-Protokoll

Wenn Sie nicht nur ASCII-Texte, sondern auch beliebige andere Dateien übertragen wollen, werden Sie nicht umhinkommen, ein genormtes Übertragungsprotokoll zu benutzen. Das XModem-Protokoll wurde 1979 von Ward Christensen in den USA eingeführt. Obwohl es heute längst durch Protokolle wie Y- und ZModem überholt ist, ist es noch weltweit Standard und wird durch fast jede Mailbox unterstützt. Wenn Sie sich ein eigenes Terminalprogramm schreiben möchten, sollten Sie zumindest dieses Übertragungsprotokoll einbauen.

Wie arbeitet XModem?

Das XModem-Protokoll arbeitet mit positiven und negativen Bestätigungen. Alle Fehler werden maximal 10mal wieder-

holt. Der Empfänger hat ein "Timeout" von 10 Sekunden, das heißt: wurde in den letzten 10 Sekunden nichts empfangen, wird ein NAK (ASCII 15H) zum Sender zurückgeschickt. Nach dem Empfang des ersten Zeichens eines Blocks arbeitet das Empfangsprogramm mit einem 1-Sekunden-Timeout für jedes einzelne Zeichen. Der Sender erwartet nach jedem gesendeten Block eine Bestätigung ACK (ASCII 06H). Kommt statt dessen ein NAK (negative Bestätigung), wird der betreffende Block nochmals gesendet (bis maximal 10mal). Ein einzelner Block ist wie folgt aufgebaut (gesendet wird mit 8 Bit, keine Parität):

<SOH> <BLOCK NR> <BLOCK NR COMPLEMENT> <...DATEN...> <chk>

wobei SOH (start of header) = ASCII 01H ist. "BLOCK NR" ist eine ein Byte lange Nummer, startet mit 0 und wird mit jedem Block um 1 erhöht. Nach 256 gesendeten Blöcken beginnt die Blocknummer wieder mit 0. "BLOCK NR COMPLEMENT" ist das Einerkomplement der Blocknummer, das sich aus 255 minus Blocknummer errechnet. "...DATEN..." sind die eigentlichen Daten in ihrer binären Form ohne jegliche Umwandlung, jeweils 128 Bytes. Am Ende folgt eine 1-Byte-Checksumme. Sehen wir uns einige Übertragungen mit Übertragungsfehlern an (Tabelle 3).

SENDER		EMPFÄNGER
(Wartet auf 1. NAK vom Empfänger) Sek	<-	<NAK> (Empfänger hat Timeout nach 10 Sekunden)
<SOH> 01 FE -DATEN- <chk>	->	(1. Block gesendet)
	<-	<ACK> (korrekt empfangen)
<SOH> 02 FD -DATEN- <chk>	->	(2. Block gesendet)
	<-	<ACK> (korrekt empfangen)
<SOH> 03 FC -DATEN- <chk>	->	(3. Block gesendet)
	<-	<NAK> (Übertragungsfehler aufgetreten)
<SOH> 03 FC -DATEN- <chk>	->	(3. Block wird wiederholt)
	<-	<ACK> (jetzt war's korrekt)
<SOH> 04 FB -DATEN- <chk>	->	(4. Block gesendet)
	<-	<ACK> (korrekt empfangen)
<ACK> ging verloren, deshalb wird nach Timeout nochmals gesendet.		
<SOH> 04 FB -DATEN- <chk>	->	(4. Block nochmals)
	<-	<ACK> (Block wird ignoriert, weil er bereits korrekt angekommen ist, ACK senden, damit der Sender weitermacht)
<SOH> 05 FA -DATEN- <chk>	->	(5. Block gesendet)
	<-	<ACK> (korrekt empfangen)
:		:
<EOT>	->	(Ende der Übertragung)
	<-	<ACK> (Bestätigung)

wobei

<SOH> = ASCII 01H [start of header]
<ACK> = ASCII 06H [acknowledge]
<NAK> = ASCII 15H [negative acknowledge]
<EOT> = ASCII 04H [end of transmission]
<chk> = Checksumme, [(b0+b1+b2+b3+...+b127) MODULO 256]

Tabelle 3: Das Protokoll einer XModem-Checksum-Übertragung

Da diese Methode allerdings etwas veraltet ist, sollte man zumindest schon ein XModem mit 16-Byte-CRC-Checksummenberechnung ("Cyclic redundancy check") einbauen. Für die Berechnung der Checksumme präsentiere ich Ihnen sogar einen C-Quelltext. XModem CRC ist grundsätzlich so aufgebaut wie das normale XModem, allerdings bestehen doch schon einige kleine Unterschiede. Wie ist das Protokoll grundsätzlich aufgebaut?

Jeder Block bei XModem CRC sieht so aus:

<SOH> <BLOCK NR> <255-BLOCK NR> <-DATEN-> <CRC low> <CRC high>

dabei sind

<SOH>	= ASCII 01H
<BLOCK NR>	= Nummer zwischen 01H und 0FFH, jeweils um 1 erhöht, wrapt nach 0FFH wieder auf 1 (nicht auf 0!)
<255-BLOCK NR>	= das Einerkomplement der Blocknummer
<CRC low>	= das untere Byte der Checksumme
<CRC high>	= das obere Byte der Checksumme

```
01:  /* update CRC */
02:  unsigned short
03:  updcrc(c, crc)
04:  register c;
05:  register unsigned crc;
06:  {
07:    register count;
08:
09:    for (count=8; —count>=0;) {
10:            if (crc & 0x8000) {
11:                    crc <<= 1;
12:                    crc += (((c<<=1) & 0400)  !=  0);
13:                    crc ^= 0x1021;
14:            }
15:            else {
16:                    crc <<= 1;
17:                    crc += (((c<<=1) & 0400)  !=  0);
18:            }
19:    }
20:    return crc;
21:  }
```

Listing 1: Die Berechnung der CRC-Checksumme in C

Um die 16-Bit-CRC-Checksumme zu berechnen, werden die Daten-Bits als Koeffizienten eines Polynoms genommen. Dieses Polynom wird zuerst multipliziert mit X^16 und dann durch das

Generatorpolynom (X^16 + X^12 + X^5 + 1) dividiert. Der übrigbleibende Rest nach der Division ist die gewünschte Checksumme. "Grummel", werden Sie jetzt sagen, das kann er sich an den Hut stecken. Recht haben Sie, und deshalb präsentiere ich Ihnen auch noch die Checksummenberechnung in C (siehe Listing 1).

Nun müssen die beiden XModems nur noch unterschieden werden. Damit der Empfänger auch weiß, daß am anderen Ende ein Programm ist, das die CRC-Checksumme berechnen kann, wird mit einem <C> statt einem <NAK> initialisiert. Was passiert nun, wenn die eine Seite CRC-Berechnungen beherrscht (der Empfänger), der Sender jedoch nicht? Wenn beide Seiten CRC-Checksummen unterstützen, sieht die Übertragung folgendermaßen aus (siehe Tabelle 5).

Hier noch einige Programmiertips, damit die Routinen nachher auch einfach anzusprechen sind und gut funktionieren.

- Die Byte-Empfangsroutine sollte mit einem Parameter aufgerufen werden, der angibt, wie lang das Timeout in Sekunden ist. Der Empfänger sollte die Routine erst mit 10 Sekunden aufrufen, dann ein <NAK> senden und das ganze 10mal versuchen.

- Nachdem das <SOH> empfangen wurde, sollte der Empfänger ein Timeout von 1 Sekunde einstellen.

- Wenn der Empfänger ein <NAK> senden möchte, sollte er vorher eine "Purge"-Routine aufrufen, die darauf wartet, daß der Empfangspuffer leer ist. Die beste Methode zu "purgen" ist, die Empfangsroutine so lange mit einem 1-Sekunden-Timeout anzuspringen, bis ein Timeout eintrifft. Erst dann sollte das <NAK> gesendet werden.

SENDER		EMPFÄNGER
	<-	<C>
		(Timeout nach 3 Sekunden)
	<-	<C>
		(Timeout nach 3 Sekunden)
	<-	<C>
		(Timeout nach 3 Sekunden)
	<-	<C>
		(Timeout nach 3 Sekunden)
	<-	<NAK>
<SOH> 01 FE -DATEN- <chk>	->	
	<-	<ACK>
usw., entsprechend wie in Tabelle 3		

Tabelle 4: Empfänger mit CRC-Option, Sender nicht

SENDER		EMPFÄNGER
	<-	<C>
<SOH> 01 FE -DATA- <xxxx>	->	
	<-	<ACK>
<SOH> 02 FD -DATA- <xxxx>	->	(Übertragungsfehler)
	<-	<NAK>
<SOH> 02 FD -DATA- <xxxx>	->	
	<-	<ACK>
<SOH> 03 FC -DATA- <xxxx>	->	
(ACK geht verloren)	<-	<ACK>
		(Timeout nach 10 Sekunden)
	<-	<NAK>
<SOH> 03 FC -DATA- <xxxx>	->	
	<-	<ACK>
<EOT>	->	
	<-	<ACK>
<xxxx> ist die CRC-Checksumme.		

Tabelle 5: Sender und Empfänger verfügen über die CRC-Option

Haben Sie Fragen, Probleme, Ergänzungen oder Vorschläge? Schreiben Sie uns:

MAXON Computer GmbH
Redaktion ST-COMPUTER
DFÜ-Ecke
Industriestraße 26
6236 Eschborn

MP / Michaela Schöbel

Literatur:

- *XModem/YModem Protocol Reference, Chuck Forsberg, 9.11.1986*
- *zmdm.doc v1.2, Jwahar Bammi, 1987*
- *K. Mulder: DFÜ selbstgemacht, Computer Persönlich 11/85, Seite 53*



Indiana Jones und der letzte Kreuzzug gibt es jetzt als Adventure von **Lucasfilm Games**. Komplett in Deutsch mit herrlichen Grafiken und animierten Filmszenen. Natürlich mit dem spielerfreundlichen Lucasfilm-Adventure-System, das so herrlich einfach zu bedienen ist. Befehl auswählen, Gegenstand oder Person anklicken, und schon macht der Computer, was man will. Kinderleicht! Deshalb auch für Anfänger geeignet.

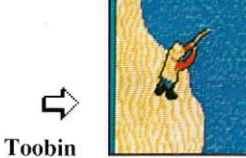




Indiana Jones

In Toobin flitzen ein oder zwei Spieler mit einem Gummireifen einen reißenden Fluß herab und versuchen, möglichst vielen Hindernissen auszuweichen. Nicht außergewöhnlich, aber wenigstens für einige Stunden ganz unterhaltsam.





Toobin

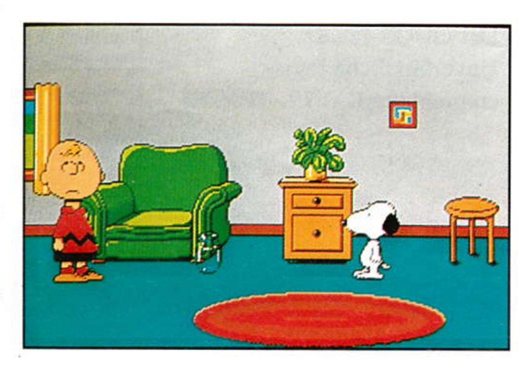
Fans von Automatenhits wird vielleicht **U.S. Golds** Spielsammlung Winners mit **Thunder Blade**, **LED Storm**, **Blasteroids** und **Impossible Mission** interessieren.

New York, London und Paris. In den U-Bahnstationen herrscht Angst und Schrecken. Straßenbanden terrorisieren friedliche Passagiere. Deshalb macht sich ein Einzelkämpfer auf den Weg, um Rowdies und Schläger durch Fausthiebe und Tritte kräftig aufzumischen. Grafik und Sound sowie Spielbarkeit sind bei **Fallen Angel** ganz passabel.

Ocean schnürte unserer Meinung nach das attraktivste Spielepaket: **Voyager**, ***International Karate+***, **Bio Challenge** und **R-Tye**. Sieht man vom mittelmäßigen "Bio Challenge" ab, erhält der Käufer drei absolute Spitzenspiele, die einst ganz oben in den Top Ten waren.

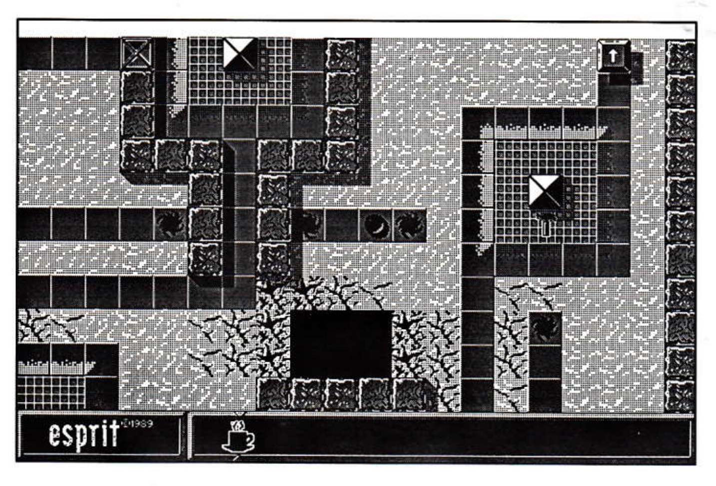
Hewson hat eine neue Compilation zu vermelden: ***Premier Collection II.*** Für knapp 100 Märker gibt es **Eliminator**, **Mercenary**, **Backlash** und **Custodian**.

Snoopy von The Edge weist eine starke Grafik auf. Alle Charaktere des erfolgreichen Comics sind haargenau getroffen. Ziel des Spiels ist es, eine verlorene Kuscheldecke zu finden. Dazu läuft Snoopy durch die Gegend, hebt Gegenstände auf und benutzt sie an der richtigen Stelle. Leider passiert bei der Suchaktion viel zu wenig, so daß dem Spieler schnell vor Langeweile die Äuglein zufallen.



Snoopy

Haben Sie Esprit ?



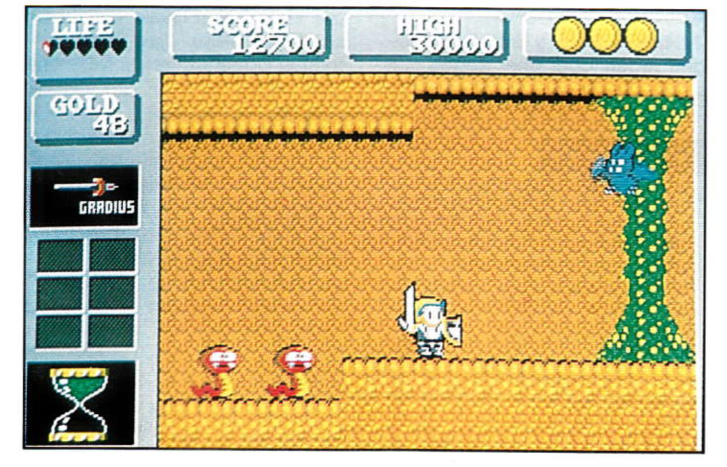
Alle kleinen und großen Kinder, die auf der ATARI-Messe in Düsseldorf auf die Jagd nach Aufklebern gingen, bekamen am Stand von Application Systems einen mit dieser philosophischen Frage in die Hand gedrückt. Für alle, die sich darüber den Kopf zerbrochen haben (denn sicherlich waren nicht Hosen und Pullover gemeint), wird nun das Geheimnis gelüftet. Esprit gibt es jetzt zu kaufen, allerdings nur in Form eines neuen Ballspiels von ASH. Hierbei handelt es sich nicht um eine Bolo-Fortsetzung, sondern eher um eine Mischung aus Memory und Marble Madness. Mit einer mausgesteuerten Kugel muß man versuchen je zwei identische Symbole, die unter bestimmten Feldern versteckt liegen, zu finden. Um keine Langeweile aufkommen zu lassen, sind von Level 1 - 100 auf dem Weg zu den Feldern immer mehr Hindernisse eingebaut So gibt es z.B. Totenkopffelder, denen man besser nicht zu nahe kommt, Bumper, Magnete und Glatteis, die die Kugel aber leider genau in Richtung Totenkopf lenken. Man findet Einbahnstraßen, Abgründe und auch schnelle Geschoße, die einem das Spielen nicht gerade erleichtern. Als kleine Abwechslung gibt es aber auch immer mal ein Meditationsbild, das etwas anders zu lösen ist (mehr sei hier nicht verraten). Man sollte sich also erst einmal jedes Bild genau ansehen, bevor man die Maus bewegt. Um das 100. Level trotz all dieser kleinen Gemeinheiten überhaupt jemals erreichen zu können, sind aber auch ein paar kleine Hilfen eingebaut, z.B. Regenschirme, die einen vor bestimmten Gefahren schützen können, Notizzettel mit Hinweisen sowie Sprengsätze und Hämmer, mit denen man sich den Weg zu den Symbolfeldern bahnen kann. *Kleiner Tip:* Man sollte sich jeden Notizzettel auch immer genau ansehen, denn manchmal gibt es statt einer dummen Bemerkung auch wichtige Hinweise oder nach einigen Leveln sozusagen als Belohnung eine achtstellige Geheimzahl, die allerdings nur einmal über den Bildschirm läuft, also immer Papier und Bleistift neben die Maus legen. Mit Hilfe dieser Nummer hat man die Möglichkeit nach Verlust aller Kugeln immer wieder in diesem Level einzusteigen. Esprit ist ein abwechslungsreiches Geduldsspiel mit einem lustigen Sound, mit dem man schnell einige Stunden verbringt. Eine gut gelungene Maussteuerung überzeugt schnell jeden geübten Spielefreund, der nicht immer nur ballern oder Kunststükke mit seinem Joystick vollbringen will.

Martina Pfahl


ESPRIT
Kategorie: Geschicklichkeit
Besonderheiten:
Spieler: 1
Monitor: monochrom
Steuerung:
Hersteller: Application Systems
Info bei: s.o.
Wertung:

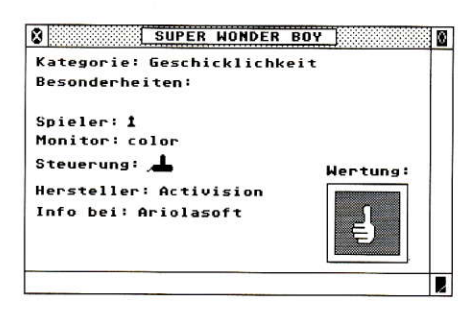
Super Wonder Boy

Im Wunderland herrschen Angst und Schrecken. Ein feuerspeiender Drachen namens Meka terrorisiert die Bewohner. Tom Tom nimmt es mit dem Drachen und seinen bösen Ungeheuern auf, damit wieder Friede im Lande einkehrt. Natürlich kann sich der kleine Wunderknabe nicht alleine auf den beschwerlichen Weg machen. Ein ST-Spieler bestimmt mit Hilfe eines Joysticks Tom Tom's Geschicke. Alleine ist es in der Wunderwelt viel zu gefährlich. Da gibt es nämlich bissige Fledermäuse, Skelette und giftige Schlangen. Trotzdem wagt sich Tom Tom völlig unbewaffnet ins Abenteuer. Er weiß schließlich, daß er in der Wunderwelt auf viele Geschäfte trifft, wo es Extrawaffen und wertvolle Tips von den Ladenbesitzern zu kaufen gibt. Gleich im ersten Shop bekommt Wonderboy ein Schwert und Medizin gratis. Alle anderen Extrawaffen und Tips ("Wo ist der Schlüssel zum nächsten Level?") bezahlt der Wunderknabe mit barer Goldmünze. Doch woher die Talerchen nehmen und nicht stehlen? Ganz einfach. Hat Tom Tom



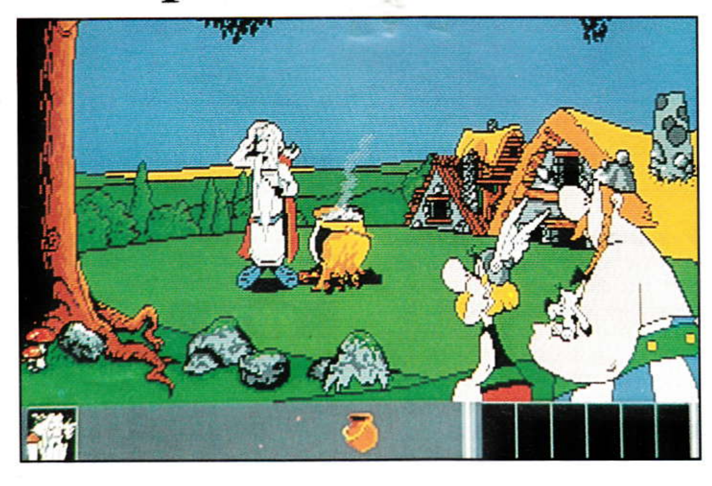
manche Monster mit seinem Schwert zur Strecke gebracht, purzeln Goldtaler durch die Gegend. Die sollte der Held einsammeln, um im nächsten Geschäft genügend Geld dabeizuhaben. Braucht man eine Extrawaffe dringend, um weiterzukommen, hat aber nicht genügend Geld dabei, kann man nachher nochmal zurücklaufen und den Gegenstand kaufen. Um einen Shop zu betreten, tritt man vor die Tür, drückt die Space-Taste, und Wonderboy pocht dagegen. Sie öffnet sich, und man sieht den Verkauf sowie Extrawaffen in einem Menü mit Preisangabe versehen. Tom Tom kann mit einem Pfeil das gewünschte Objekt aussuchen und kaufen - vorausgesetzt auf seinem Goldtalerkonto ist genügend Geld vorhanden. Ansonsten ist der Held damit beschäftigt, den Ausgang zum nächsten Level zu erreichen. Dazu muß er einen Haufen Monster niederstechen, auf Plattformen und Aufzüge springen und viele Tips und Extrawaffen kaufen. "Super Wonder Boy" wird an keiner Stelle langweilig. Activisions neues Geschicklichkeitsspiel hat mindestens genauso viel Klasse wie Giana Sisters. Neben ausgezeichneter Spielbarkeit zeichnet sich das Spiel auch durch ein fast ruckelfreies Scrolling, farbenprächtige Grafiken sowie eine pfiffige Melodie von David Whittacker aus. Fans von Geschicklichkeitsspielen kommen an "Super Wonder Boy" nicht vorbei.

CBO


SUPER WONDER BOY
Kategorie: Geschicklichkeit
Besonderheiten:
Spieler: 1
Monitor: color
Steuerung:
Hersteller: Activision
Info bei: Ariolasoft
Wertung:

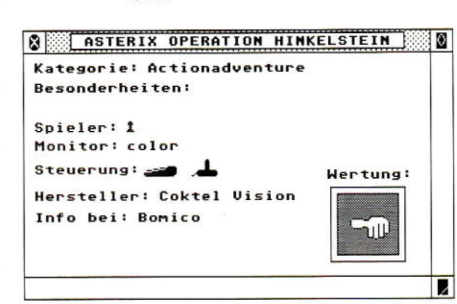
Asterix: Operation Hinkelstein

Schon seit zwei Monaten strömt jung und alt in die Kinos und will ihn sehen: Asterix, den pfiffigen Comic-Gallier, der vor der Wildschweinjagd so gerne eine Horde Römer aufmischt. "Coktel Vision" hat sein neuestes Filmabenteuer "Operation Hinkelstein" als Computerspiel umgesetzt. Zu Beginn zeigt der ST eine animierte Sequenz zur Einführung in die Handlung: Ein Hinkelstein fliegt durch die Luft und trifft den Druiden Miraculix direkt am Kopf. Dabei wollte Obelix doch einen Römer erledigen. Ein peinlicher Fehlwurf! Nur der Dorfdruide ist nämlich in der Lage, den Zaubertrank zu brauen, der die Gallier unbesiegbar macht. Ohne das Gesöff hätte sich das Dorf nicht so lange gegen die übermächtigen Römer behaupten können. Ein Drama! Der Druide hat durch den enormen Schlag auf den Hinterkopf sein Gedächtnis verloren und sämtliche Rezepturen, ja sogar seinen eigenen Namen vergessen. Jetzt kommt Asterix ins Spiel. Er muß für den kopfkranken Druiden Kräuter sammeln, einen Heiltrank brauen und ihn Miraculix einflößen. Gesteuert wird der Held durch Tastatur oder Joystick. Das Abenteuer beginnt



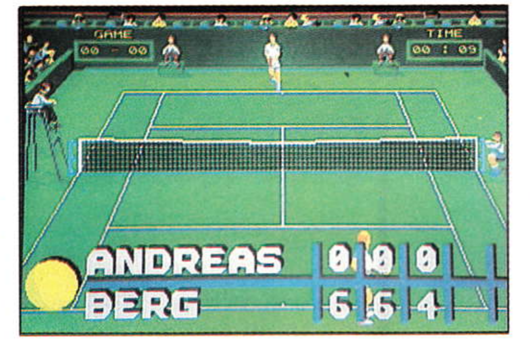
im gallischen Dorf. Dort kann Asterix an einigen Ständen Gegenstände kaufen und auch die gefundenen Zutaten zu einem Trank zusammenmischen. Bevor man Miraculix davon zu trinken gibt, sollte man allerdings einen Römer fangen und ihn als Versuchskaninchen einsetzen. Es gibt nämlich 20 verschiedene Möglichkeiten, mit den Zutaten einen Trank zu brauen. Und nur eine davon wird Miraculix heilen. Um im Dorf alle nötigen Gegenstände kaufen zu können, braucht man viele Sesterzen. Doch woher nehmen und nicht stehlen? Aus dem Römerlager! Um an Geld zu gelangen, muß Asterix nicht den Römern die Sesterzen aus den Taschen prügeln. Es reicht, in einem Würfelspiel zu gewinnen, das die Soldaten im Lager veranstalten. Mit mehr Punkten als der Gegner gewinnt Asterix einen Sesterzenhaufen. Das Geld ist allerdings sekundär. Viel wichtiger sind die Heilkräuter, die vorwiegend im Wald zu finden sind. Aber Vorsicht! Dort treiben sich prügelfreundliche Römer herum, die Asterix gesundheitlich schwächen, wenn er ihnen nicht ein paar ordentliche Backpfeifen verpaßt. Manchmal trifft er auch auf Wildschweine. Für jede erlegte Sau hagelt es Punkte. Alles in allem ist Coktel Visions neues Asterixspielchen ganz hübsch. Die Grafiken halten sich originalgetreu an die Comic-Vorlagen, und die Animationen sind witzig, allerdings hapert's beim Sound. Eine kurze digitalisierte Titelmelodie und einige Geräuscheffekte während des Spielverlaufs sind alles, was der Spieler zu Gehör bekommt. Spielerisch ist "Operation Hinkelstein" nichts Besonderes. Herumlaufen, Kräuter sammeln, würfeln und ab und zu ein paar Römer verdreschen ist nicht gerade sehr abwechslungsreich. Deshalb versteht das Spiel auch nur für wenige Stunden zu unterhalten.

CBO



ASTERIX OPERATION HINKELSTEIN
Kategorie: Actionadventure
Besonderheiten:
Spieler: 1
Monitor: color
Steuerung:
Hersteller: Coktel Vision
Info bei: Bomico
Wertung:

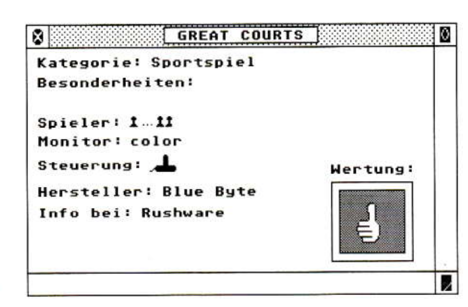
Great Courts



Einmal in der Weltrangliste ganz oben stehen...". Davon träumen viele Tennisfreaks, die tagtäglich auf bundesdeutschen Plätzen ihren Idolen Boris Becker und Steffi Graf nacheifern. Dank "Blue Byte" lassen sich derartige Ambitionen jetzt wenigstens am ST verwirklichen. Das Game führt nämlich in die Welt des Tennissports. "Great Courts" verwaltet Daten von 63 Tennisspielern in einer Weltrangliste. Sie sind Nummer 64 und müssen sich durch Siege bei den vier Grandslam-Turnieren in Melbourne, Paris, Flushing Meadow und Wimbledon an die Spitze spielen. Damit sie bei den Wettkämpfen nicht schon in der ersten Runde herausfliegen, besteht die Möglichkeit, einen Trainingsmodus in Anspruch zu nehmen. Da können Sie Aufschläge üben, gegen Ballmaschinen antreten oder ein Match gegen einen Mitmenschen wagen. Dies dient dazu, um Joystick-Steuerung und Timing in den Punktspielen zu beherrschen. Je mehr Gegner Sie schlagen und je mehr Turniere Sie gewinnen, desto höher steigen Sie in der Weltrangliste. Das Tolle ist, jeder Computergegner in der Weltrangliste existiert wirklich und besitzt im Computerspiel dieselbe relative Spielstärke wie in der Realität. Um Blondschopf Boris Becker zu besiegen, muß man schon ganz schön fit sein, für Ivan Lendl ein Meister. Mit Hilfe des Joysticks schlagen Sie Aufschläge, Schmetterbälle, Top Spin, Slice und Lops.

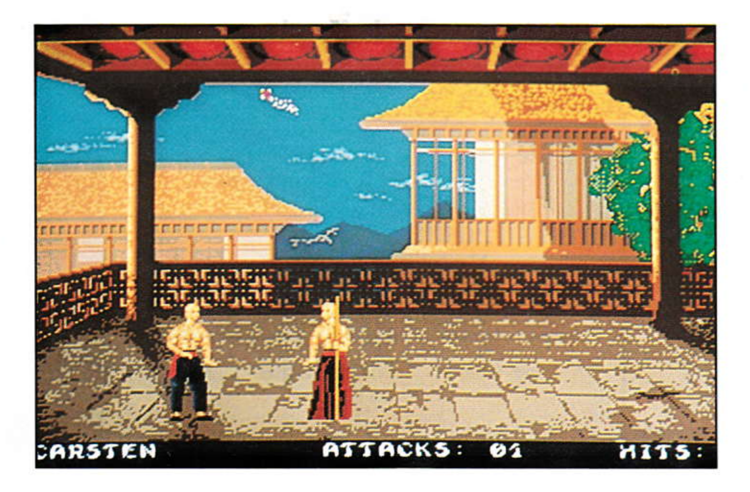
Alle Schläge und Bewegungen laufen in flüssigen Animationen ab. Spielerisch versteht "Great Courts" ebenfalls zu begeistern. Es macht unheimlich viel Spaß. Da stimmt einfach alles. Der Ball fliegt nach physikalischen Gesetzen (das ist bei Programmen dieser Art leider nicht immer selbstverständlich), und der Schwierigkeitsgrad während eines Turniers steigt automatisch beim Halbfinale. Dann bekommt man einen Aufschlag nur übers Netz, wenn man den entsprechenden Schwierigkeitsgrad zuvor im Trainingsmodus geübt hat. Um auf Platz 1 der Weltrangliste zu kommen, muß man sechs Grand Slam-Turniere gewinnen. Wer nicht gern allein spielt, kann auch im Zweispielermodus gegen einen Mitspieler antreten und sich spaßige Partien liefern. "Great Courts" ist das beste Tennisspiel auf dem Markt.

CBO



GREAT COURTS
Kategorie: Sportspiel
Besonderheiten:
Spieler: 1...2
Monitor: color
Steuerung:
Hersteller: Blue Byte
Info bei: Rushware
Wertung:

Chambers of Shaolin



Ein gewisser Hang Foy Qua vermißt sein Schwesterlein. Schergen des Kaisers haben das Mädel entführt und halten es irgendwo gefangen. Bevor sich der Held auf die Socken macht, um sein Schwesterlein zu befreien, sollte er seinen Körper in den sechs Kammern des Shaolin trainieren. Nur als ausgebildeter Kung Fu-Kämpfer stehen seine Chancen günstig, nicht von den Wächtern zu Hackfleisch verarbeitet zu werden. In der ersten Kammer steht der Lehrmeister und schwingt seinen Kampfstock. Es gilt, möglichst vielen Attacken auszuweichen. Fliegende Hackebeile und Feuerkugeln stehen beim zweiten Training in der "Kammer der Behendigkeit" auf dem Programm. Sie müssen einen Flickflack springen, sich ducken, über die Hindernisse hüpfen, um nicht verletzt zu werden. Danach folgt die dritte Kammer des Shaolin. Es handelt sich hierbei um einen Test der Sprungstärke. Hierzu muß Hang Foy Qua auf vier sich auf- und abbewegenden Pfählen hin- und herspringen, stets zu dem Pfahl, an dessen unterem Ende sich eine Markierung befindet. Springt er nicht im richtigen Timing, plumpst er ins Wasser. In der vierten Kammer sind schnelle Tritte gefragt. Eine Eisenkugel hängt an einem Seil von der Decke und muß durch Tritte in Schwingung gebracht werden. Die pendelnde Kugel soll einen Mechanismus treffen, der zwei Wasserrohre verschließt. Gelingt dies nicht, ertrinkt der Held jämmerlich. Mit einem gezielten Schlag versucht Hang Foy Qua, in der "Kammer der Stärke" Holzbretter zu zerschlagen. Erst muß die Hand positioniert werden, durch schnelles Hin- und Herrütteln des Joysticks vergrößert sich die Schlagkraft. Bei der sechsten und letzten Prüfung steht ein Lehrmeister in einem Boot und wirft Feuerkugeln auf eine Holzbrücke. Dort steht Hang Foy Qua und strengt sich an, alle Kugeln durch Fußfeger von der Brücke zu kicken. Alle sechs Prüfungen haben ein Ziel: der Spieler soll die Eigenschaften und Talente seines Kämpfers selber formen. In jeder Kammer werden andere Fertigkeiten und Schläge trainiert, die dem Fighter nachher in den Zweikämpfen bei der Befreiungsaktion zugute kommen. Hat ein Fighter alle sechs Kammern des Shaolin absolviert, können Sie seine Werte für Angriffs-, Defensivstärke, Ausdauer und körperliche Verfassung auf eine Charakter-Disk speichern. Vor jeder Kampfphase gegen die Wächter verlangt der ST die Charakter-Disk und lädt die spezifischen Daten des edierten Kämpfers in den Speicher. Wer sich in den Kammern des Shaolin nicht angestrengt hat, wird merken, daß beispielsweise die Energieanzeige schon zu Beginn des Kampfes sehr niedrig ist. Da braucht der Gegner nur einen Schlag zu plazieren, und der Kampf ist für Hang Foy Qua beendet. Sie merken, "Chambers of Shaolin" ist nicht so primitiv, wie die meisten anderen asiatischen Kampfsportspiele, die es sonst noch auf dem Markt gibt. Nicht nur die Konzeption von Thalions neuem Game stellt etwas Besonderes dar. Auch die technische Ausführung ist einzigartig.

CBO



In der nächsten ST-Computer lesen Sie unter anderem

WordPerfect

Eigentlich war der Test der neuesten WordPerfect-Version, die jetzt ja über ATARI vertrieben wird, schon für diese Ausgabe geplant gewesen. Doch wie es nun mal so kommt, wurde noch ein Programmfehler von der WordPerfect GmbH festgestellt, so daß wir den Test verschieben mußten. Mit Sicherheit sind Sie genauso auf WordPerfect gespannt wie wir; ist es doch die Standard-Textverarbeitung auf den MS-DOS-Rechnern.

Neue Drucker

In der nächsten Ausgabe wollen wir Ihnen zwei neue Drucker vorstellen, die jetzt auf den Markt gekommen sind. Es handelt sich dabei um den neuen Billigdrucker von NEC, den P2plus, der den P2200 ablösen soll, und den neuen STAR XB24-10.

Mortimer, der Butler

Schon auf der ATARI-Messe stellte sich Mortimer, der Butler, aus dem Hause OMIKRON. den neugierigen Besuchern vor. Sein großer Vorteil ist, immer wenn man ihn benötigt, braucht man nur eine Tastenkombination zu drücken und schon stehr er Ihnen zu Diensten. Als fleißiger Diener hilft er Ihnen bei der alltäglichen Arbeit mit dem ST. Lassen Sie sich überraschen, ob Mortimer durch eine gute Schule gegangen ist und seinen Knigge beherrscht.

Die nächste ST Computer erscheint am Fr., den 26.1.90

Fragen an die Redaktion

Ein Magazin wie die ST-Computer zu erstellen, kostet sehr viel Zeit und Mühe. Da wir ja weiterhin vorhaben, die Qualität zu steigern, haben wir Redakteure ein großes Anliegen an Sie, liebe Leserinnen und Leser:

Bitte haben Sie Verständnis dafür, daß Fragen an die Redaktion nur Donnerstags von 14^{00}-17^{00} Uhr telefonisch beantwortet werden können.

Vielen Dank für Ihr Verständnis

Impressum ST Computer

Chefredakteur: Uwe Bärtels (UB)
Stellvertreter: Harald Egel (HE)

Redaktion:
Uwe Bärtels (UB)
Harald Egel (HE)
Harald Schneider (HS)
Martin Pittelkow (MP)

Redaktionelle Mitarbeiter:
C.Borgmeier (CBO)
Claus Brod (CB)
Ingo Brümmer (IB)
Derek dela Fuente (ddF)
Stefan Höhn (SH)
Raymund Hofmann (RH)
Dieter Kühner (DK)
Jürgen Leonhard (JL)
Claus P. Lippert (CPL)
Markus Nerding (MN)
Chr. Schormann (CS)
R.Tolksdorf (RT)

Autoren dieser Ausgabe:
P.Denk
G.Gehnen
U.Hax
A.Hill
T.Huber
A.Kohler
M.Müller
P.Neuchel
M.Pfahl
D.Rabich
B.Rosenlecher
U.A.Ruttkamp
U.Seimet
U.Thürmann
T.Werner
R.Wiechert
R.Wiesler

Auslandskorrespondenz:
C.P.Lippert (Leitung), D.Dela Fuente (UK)

Redaktion: MAXON Computer GmbH
Postfach 59 69
Industriestr. 26
6236 Eschborn
Tel.: 0 61 96/48 18 14, FAX : 0 61 96/4 11 37

Verlag: Heim Fachverlag
Heidelberger Landstr. 194
6100 Darmstadt 13
Tel.: 0 61 51/5 60 57, FAX : 0 61 51/5 56 89 + 5 60 59

Verlagsleitung:
H.J.Heim

Anzeigenverkaufsleitung:
U.Heim

Anzeigenverkauf:
K.Margaritis

Anzeigenpreise:
nach Preisliste Nr.3, gültig ab 1.1.88
ISSN 0932-0385

Grafische Gestaltung:
Gabriele Imhof

Layout:
Kerstin Feist, Manfred Zimmermann

Titelgestaltung:
Gunter Wenzel (Tel.: 06172/37193)

Fotografie:
Gabriele Imhof, Archiv, K.Ohlenschläger

Produktion:
K.H.Hoffmann, B.Kissner

Druck:
Frotscher Druck GmbH

Lektorat:
V.Pfeiffer

Bezugsmöglichkeiten:
ATARI-Fachhandel, Zeitschriftenhandel, Kauf- und Warenhäuser oder direkt beim Verlag

ST Computer erscheint 11 x im Jahr
Einzelpreis: DM 7,-, ÖS 56,-, SFr 7,-
Jahresabonnement: DM 70,-
Europ. Ausland : DM 90,- Luftpost : DM 120,-
In den Preisen sind die gesetzliche MWSt. und die Zustellgebühren enthalten.

Manuskripteinsendungen:
Programmlistings, Bauanleitungen und Manuskripte werden von der Redaktion gerne angenommen. Sie müssen frei von Rechten Dritter sein. Mit seiner Einsendung gibt der Verfasser die Zustimmung zum Abdruck und der Vervielfältigung auf Datenträgern der MAXON Computer GmbH. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte wird keine Haftung übernommen.

GFA für ATARI

GFA-BASIC

Weltweit über 100 000mal im Einsatz!

- **GFA-BASIC 3.0 EWS ST**
  Hochgeschwindigkeitsinterpreter und integrativer Compiler in einem als komplettes Entwicklungssystem. Einbindung von Assembler und C-Source-Codes in GFA-BASIC-Programme **DM 198,–**
- **GFA-BASIC 2.0 EWS ST**
  Das GFA-BASIC 2.0 Entwicklungssystem ST. Interpreter + Compiler für Einsteiger. (Upgrade-Möglichkeit zum GFA-BASIC 3.0 Entwicklungssystem ST DM 160,–) **DM 49,90**
- **GFA-GUP GEM UTILITY-PACKAGE** **DM 149,–**
- **GFA-BASIC KONVERTER nach C** **DM 498,–** neu

GFA-ASSEMBLER ST

Professioneller Makro-Assembler für 68 000-Programmierer: Leistungsfähiger Editor mit integriertem Assembler und Linker. Nachladbarer Debugger **DM 149,–**

GFA-BÜCHER

- **GFA-BASIC 3.0 ST Training** Der ideale Einstieg in die Version 3.0 mit 14 Themenschwerpunkten. 272 Seiten, Hardcover, ISBN 3-89317-005-7 **DM 29,–**
- **GFA-BASIC ST: Version 3.0** Das Umsteigerbuch 394 Seiten, Hardcover, inkl. Diskette, ISBN 3-89317-004-9 **DM 59,–**
- **GFA-BASIC Programmierung** Programmierhilfe von der Idee, zum Entwurf, zum Programm. Ca. 300 Seiten, Hardcover, inkl. Diskette ISBN 3-89317-003-0 **DM 49,–**
- **GFA-BASIC-Buch Frank Ostrowski (ST)** Frank Ostrowski über sein GFA-BASIC (Programmoptimierung). Ca. 300 Seiten, Hardcover, inkl. Diskette ISBN 3-89317-001-4 **DM 79,–**
- **Das GFA-Anwenderburch** Wann GFA-BASIC? Wann GFA-ASSEMBLER? Die Antwort finden Sie in dem neuen GFA-Anwenderbuch Ca. 450 Seiten, Hardcover, inkl. Diskette, ISBN 3-89317-011-1 **DM 59,–** neu

GFA-DRAFT-plus ST

Leistungsfähiges, zweidimensionales CAD-Programm, seit Jahren bewährt, tausendfach im Einsatz. (Symbolbibliotheken zu GFA-DRAFT-plus auf Anfrage)

DM 349,–

neu

GFA-DRAFT-KONTAKT

Kontaktverwaltung für den gesamten Schaltplan

DM 398,–

GFA-STRUKTO

Dialogorientierte programmierte Unterweisung zum strukturierten Programmieren

DM 249,– neu

GFA-STATISTIK

Das professionelle Statistikpaket. Über 70 Verfahren der beschreibenden und schließenden Statistik. Umfangreiches Handbuch, Beschreibung jedes Verfahrens sowohl von der rein formalen als auch der Anwendungsseite **DM 998,–**
Campus- und Studentenversion: **Preis auf Anfrage.**

Anruf genügt 0211/5504-0

GFA Systemtechnik GmbH
Heerdter Sandberg 30-32
D-4000 Düsseldorf 11
Tel. 0211/5504-0 · Fax 0211/550444

